BBRv2-TI: A Utility-Guided Congestion Control Algorithm for Tactile Internet over QUIC

Muhammad Hanif Lashari, Shakil Ahmed, Wafa Batayneh, and Ashfaq Khokhar
Department of Electrical and Computer Engineering, Iowa State University, Ames, USA
{mhanif, shakil, batayneh, ashfaq}@iastate.edu

***Abstract*—Tactile Internet applications such as remote surgery and real-time haptic control require ultra-low delay, high reliability, and steady throughput. Existing congestion control algorithms (CUBIC, Reno, BBRv2) degrade under jitter and loss. This work presents BBRv2-TI for QUIC-based TI, modeling congestion control as a multi-objective optimization over throughput, jitter, and loss with a gradient-free utility. Eleven BBRv2 parameters in QUICHE are tuned to improve responsiveness without kernel or cross-layer changes. In emulation, BBRv2-TI reduces jitter by up to 98%, lowers loss by about 30%, and increases throughput by as much as 1.5× over baselines, indicating suitability for delay sensitive TI systems.**

***Index Terms*—Tactile Internet, QUIC, Congestion Control, Ultra-Low Latency, Delay-Sensitive Applications, Beyond 5G/6G.**

I. Introduction

The *Tactile Internet* (TI) enables real-time, reliable, and low-latency interaction by transmitting touch and motion through a network. Unlike traditional audio–video communication, TI supports remote actuation and haptic feedback, allowing tasks such as telesurgery, industrial automation, and robotic teleoperation [1], [2]. Because haptic signals are highly delay-sensitive, TI demands round-trip times (RTTs) below 1 ms and reliability exceeding 99.99999% [3]. Even slight jitter or packet loss can cause instability in critical applications such as remote robotic surgery [3]–[5].

Despite advances in 5G and emerging 6G networks, end-to-end transport performance often remains a bottleneck due to congestion, delay, and loss [6]. As illustrated in Fig. 1, mission-critical use cases such as telesurgery and autonomous driving occupy the ultra-reliable, low-latency region, emphasizing the need for efficient transport-layer congestion control. Among modern transport protocols, *QUIC* offers a practical foundation for TI because it supports multiplexed streams, encryption by default, and user-space deployment. However, QUIC's performance strongly depends on its congestion control algorithm (CCA). Existing CCAs—Reno, CUBIC, and *Bandwidth and Round-Trip Time (BBR)*—perform well for bulk transfers but degrade in networks with high variability. In particular, *BBRv2* improves fairness and queue control through delay-based probing [7], yet its conservative pacing and fixed parameters limit responsiveness in high-delay or lossy environments, which are typical for TI systems. This motivates an enhanced design that balances throughput, delay, and reliability without kernel modifications.

The key contributions are: BBRv2-TI is presented as a utility guided, empirically tuned variant of BBRv2 for QUIC based TI; congestion control is formulated as a multi objective optimization that jointly minimizes RTT, jitter, and loss while maintaining throughput; eleven BBRv2 parameters in Cloudflare QUICHE are tuned without kernel or cross layer changes; and emulation shows that BBRv2-TI lowers jitter and loss while sustaining comparable throughput across low, medium, and high impairment profiles.

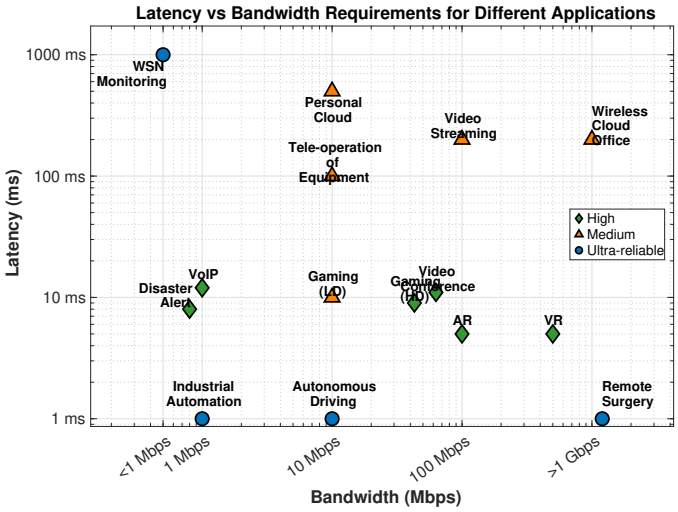

Fig. 1. Latency vs. bandwidth and reliability across applications.

II. Related Work

Congestion control algorithms can be broadly categorized according to the type of feedback they rely on, such as packet loss, delay variation, or explicit modeling of network conditions.

A. Loss-Based Congestion Control

Loss-based methods treat packet loss as a congestion signal. TCP Reno introduced the Additive Increase Multiplicative Decrease rule to regulate the sending rate [8]. TCP CUBIC uses a cubic growth function to improve scalability on high bandwidth and long delay paths [9]. These methods can create large queues, which is undesirable for very low-latency tasks.

B. Delay-Based Congestion Control

Delay-based algorithms detect congestion by monitoring increases in packet round-trip times (RTTs), allowing for earlier congestion detection. TCP Vegas adjusts the sending

rate based on RTT variations, aiming to maintain a balance between throughput and delay [10]. Copa targets low latency by adapting the sending rate to maintain a target queuing delay and demonstrates improved performance across various network conditions [11]. These algorithms offer better latency control but may suffer from fairness issues when coexisting with loss-based flows.

C. Model-Based Congestion Control

Model-based algorithms estimate network parameters to proactively adjust sending rates. BBR (Bottleneck Bandwidth and Round-trip time) estimates the bottleneck bandwidth and RTT to determine the optimal sending rate, achieving high throughput and low latency [12]. BBRv2 enhances BBR by incorporating loss signals and Explicit Congestion Notification (ECN) to improve fairness and stability. These approaches aim to optimize performance by proactively modeling network behavior.

D. Utility-Based Congestion Control

Utility-based algorithms optimize a defined utility function, balancing throughput, delay, and loss. PCC Vivace is an online-learning congestion control algorithm that adapts sending rates by optimizing a utility function based on real-time delay and loss metrics [13]. Hercules extends utility-based control by supporting heterogeneous QoS requirements through an online learning framework and is implemented as a QUIC module for real-world deployment [14]. These methods aim to dynamically align transmission behavior with application-level performance goals.

As shown in Fig. 2, the sender operates a closed-loop control system that adapts transmission based on receiver feedback. Acknowledgments from the network are processed to extract round-trip time, jitter, and loss, which guide pacing and congestion window updates. The proposed design adds a utility-guided tuning module that adjusts internal parameters to satisfy the latency and reliability goals of Tactile Internet applications such as remote surgery [15]. It is implemented in the QUICHE transport stack for user space operation, keeping the original BBRv2 pacing and phase logic while applying lightweight parameter tuning without kernel changes.

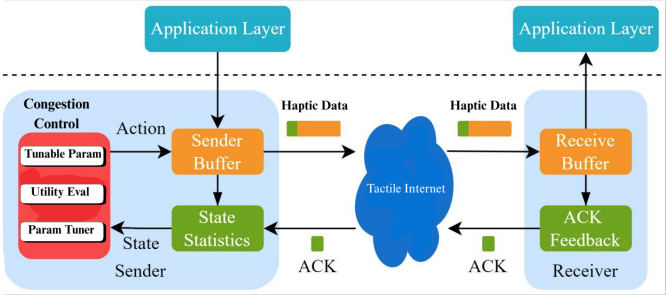

Fig. 2. A schematic overview of end-to-end congestion control architecture

III. Problem Statement

TI applications require ultra low latency (below 1 ms), high reliability, and steady throughput to support real time tasks such as remote robotic surgery. Current congestion control algorithms, including CUBIC, Reno, BBRv1, and BBRv2, are not tailored for these conditions. Although BBRv2 improves fairness through model based probing, it still faces performance loss in high delay and lossy networks because of conservative pacing and slow recovery. These effects are critical in wide area and wireless deployments typical of TI systems.

This work introduces BBRv2-TI, a tuned variant of BBRv2 in the Cloudflare QUICHE stack. The goal is to enhance BBRv2's behavior under TI constraints by adjusting a set of internal control parameters that operate during the STARTUP, DRAIN, PROBE_BW, and loss recovery phases.

Let $\mathbf{x} = [x_1, x_2, \ldots, x_{11}]$ represent the vector of tunable parameters that define BBRv2–TI. Each x_i corresponds to a control variable that influences responsiveness, pacing, or recovery. The parameters and their limits are: $x_1 = g_c \in [1.0, 3.0]$ startup congestion window gain, $x_2 = g_p \in [1.0, 3.0]$ startup pacing gain, $x_3 = \beta \in [0.5, 0.9]$ backoff factor for loss recovery, $x_4 = L_{\text{thresh}} \in [1, 8]$ loss threshold to exit STARTUP, $x_5 = q_r \in [200, 1000]$ ms probe RTT interval, $x_6 = q_d \in [5, 100]$ ms probe RTT duration, $x_7 = f_{\text{bw}} \in [0.85, 1.1]$ full bandwidth threshold, $x_8 = L_{\text{full}} \in [1, 10]$ rounds to confirm full bandwidth, $x_9 = h_i \in [1.0, 2.5]$ headroom factor for inflight limit, $x_{10} = \alpha \in [0.5, 1.5]$ amplitude of gain cycling in PROBE_BW, and $x_{11} = \delta \in [0.0, 1.0]$ smoothing factor for congestion window updates.

TABLE I
Notation Used in Optimization Model

Symbol	Description
$\mathbf{x}$	Vector of tunable parameters $[x_1, x_2, \ldots, x_{11}]$
$\text{RTT}(\mathbf{x})$	Average round trip time
rtt_i	Round trip time for packet i
$L_r(\mathbf{x})$	Packet loss ratio
$\text{Jitter}(\mathbf{x})$	Mean variation in RTT between packets
$\tau(\mathbf{x})$	Average throughput (bytes per second)
T_0	Minimum acceptable throughput threshold
w_1, w_2, w_3	Weights for RTT, loss, and jitter objectives

The performance metrics are defined as:

$$\text{RTT}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \text{rtt}_i \tag{1}$$

$$\text{Jitter}(\mathbf{x}) = \frac{1}{N-1} \sum_{i=2}^{N} |\text{rtt}_i - \text{rtt}_{i-1}| \tag{2}$$

$$L_r(\mathbf{x}) = \frac{P_{\text{lost}}}{P_{\text{sent}}} \tag{3}$$

$$\tau(\mathbf{x}) = \frac{B_{\text{ack}}}{T_{\text{transfer}}} \tag{4}$$

The tuning objective is formulated as a multi-objective optimization problem:

$$\min_{\mathbf{x}} \; w_1 \text{RTT}(\mathbf{x}) + w_2 L_r(\mathbf{x}) + w_3 \text{Jitter}(\mathbf{x}) \tag{5}$$

$$\text{s.t. } \tau(\mathbf{x}) \geq T_0 \tag{6}$$

Here $\text{RTT}(\mathbf{x})$ measures end to end latency, $L_r(\mathbf{x})$ represents the packet loss ratio, and $\text{Jitter}(\mathbf{x})$ measures delay variation. The throughput function $\tau(\mathbf{x})$ captures the achieved transfer rate. The scalar T_0 defines the minimum acceptable throughput, while w_1, w_2, and w_3 are weights that balance the three objectives. This optimization framework enables BBRv2–TI to balance responsiveness, delay, and reliability for ultra-low latency Tactile Internet applications.

IV. Proposed Approach

To solve the optimization problem, an empirical gradient-free tuning strategy is implemented in Cloudflare QUICHE. Controlled network conditions are emulated on Linux using `tc` to model delay, jitter, and loss. Inspired by PCC Vivace, a runtime feedback loop evaluates connection performance and adjusts parameters to maximize a utility function U defined as

$$U(x_i, \text{RTT}_i, L_i) = x_i t - b x_i \left| \frac{\Delta \text{RTT}_i}{\Delta t} \right| - c x_i L_i \tag{7}$$

where x_i is the instantaneous sending rate derived from pacing or CWND, t is a baseline normalization factor, $\left|\frac{\Delta \text{RTT}_i}{\Delta t}\right|$ represents the rate of RTT change (used as jitter proxy), and L_i is the observed packet loss. Constants b and c are penalty weights for jitter and loss. The objective aligns throughput maximization with penalties on delay and loss.

The utility function favors higher throughput (term $x_i t$) while penalizing variability. Parameter updates are applied iteratively, and their influence is evaluated using the resulting change in utility. The goal is to find $\mathbf{x}$ that maximizes U while satisfying the throughput constraint in (7).

A. Convergence Strategy

At each interval, let $\Delta U = U_t - U_{t-1}$ and update a subset of $\mathbf{x}$ by $x \leftarrow \text{clip}(x + \eta s \, \text{sign}(\Delta U), \text{bounds})$, with rollback if $\Delta U < 0$. Here η adapts step size and $s \in \{\pm 1\}$ is the perturbation direction. To maintain stability and avoid oscillation, three methods are used: (1) randomized local search within bounds with rollback on utility drop, (2) adaptive step sizing for gradual updates near thresholds, and (3) rolling window averaging for smoothing. These stabilize latency, jitter, and loss without reducing throughput. Updates stay within bounds in Section III and reuse the same tuned parameters across profiles.

B. Impact of Tunable Parameters

Each variable in $\mathbf{x}$ affects the behavior of BBRv2-TI. g_c and α control startup aggressiveness and gain cycling, influencing rate ramp-up. g_p, L_{thresh}, and δ tune pacing smoothness and burst mitigation. q_r and q_d set probe RTT frequency and duration. f_{bw} and L_{full} detect bandwidth plateaus and control exit from STARTUP. β and h_i manage recovery and inflight tolerance under loss. Utility is logged per iteration, and updates move toward higher utility values while maintaining throughput guarantees.

C. Comparison with Utility Based Methods

Utility-based congestion control has been explored to balance throughput, delay, and reliability. PCC Vivace [13] and Hercules [14] both define runtime utility functions but have drawbacks in real-time environments. Vivace reacts strongly to transient RTT fluctuations, causing false congestion signals and instability. Hercules improves fairness with nonlinear scaling but requires kernel coordination and per-flow state, limiting use in user-space transports such as QUIC.

BBRv2-TI combines Vivace's learning concept with a lightweight gradient-free approach optimized for empirical robustness. It avoids overfitting, adapts smoothly to noise, and maintains stable RTT slopes under varying network conditions. This makes it suitable for real-time and embedded Tactile Internet systems that need ultra-low latency and high reliability.

TABLE II
Comparison of Utility-Based Congestion Control Frameworks

Feature	Vivace [13]	Hercules [14]	BBRv2-TI
Computational Overhead	Low	High	Medium
Fairness Guarantees	None	Strong	Basic
Stability in RTT	Poor	Good	Excellent
TI Suitability	Limited	Low	Optimized
Deployment Complexity	Low	High	Low

BBRv2-TI fits directly within QUICHE's BBRv2 module without kernel modification. It supports iterative tuning during experiments, does not need cross-flow coordination, and is resilient to measurement noise. Compared to Vivace, it is steadier under variable delay, and compared to Hercules, it is simpler and deployable in real-time robotic systems.

V. Results

BBRv2–TI was implemented on Cloudflare QUICHE and evaluated on Ubuntu 20.04 (Intel i7, 32 GB RAM). Network conditions were emulated with Linux `tc` using three profiles: low impairment (5 ms delay, 1 ms jitter, 0.5% loss), medium (20 ms, 5 ms, 1%), and high (50 ms, 10 ms, 2%). A Dockerized perfSONAR Testpoint validated the settings with OWAMP and `iperf3`. Each result is the median of ten runs with 95% confidence intervals, jitter in ms and throughput in MB/s, and the same tuned parameter set was used across all profiles. Under low impairment (Fig. 3), BBRv2–TI attains 3.50% loss, 200 ms jitter, and 0.99 MB/s; BBR gives 3.74%, 300 ms, 0.90 MB/s; BBRv2 gives 3.58%, 300 ms, 0.84 MB/s; Reno reaches 1.07 MB/s but with 570 ms jitter. Under medium impairment (Fig. 4), BBRv2–TI yields 130 ms jitter, 0.67% loss, and 3.44 MB/s; CUBIC reaches 5.72 MB/s but with 6250 ms jitter and 1.27% loss; BBR and BBRv2 deliver 2.79/3.39 MB/s with higher jitter (410/340 ms) and slightly higher loss (0.66/0.87%). Under high impairment (Fig. 5), BBRv2–TI records 2.10% loss, 580 ms jitter, and 1.17 MB/s;

CUBIC is 1.23 MB/s with 1160 ms jitter and 2.29% loss; Reno drops to 0.29 MB/s. Across all settings BBRv2–TI maintains a balanced tradeoff of throughput, jitter, and loss, which better supports interactive and haptic workloads where stability is more valuable than peak rate.

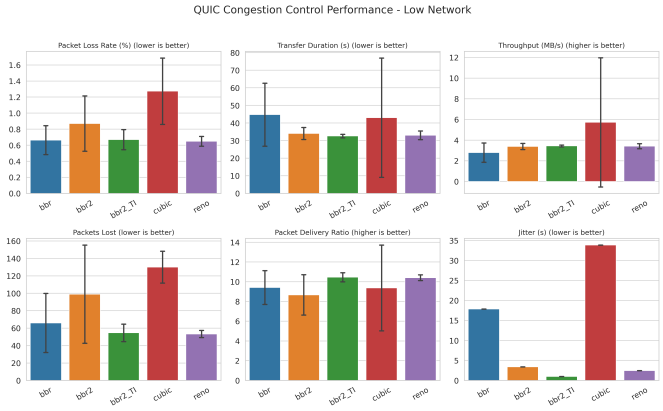

Fig. 3. Low: 5 ms delay, 1 ms jitter, 0.5% loss; n=10 median, 95% CI.

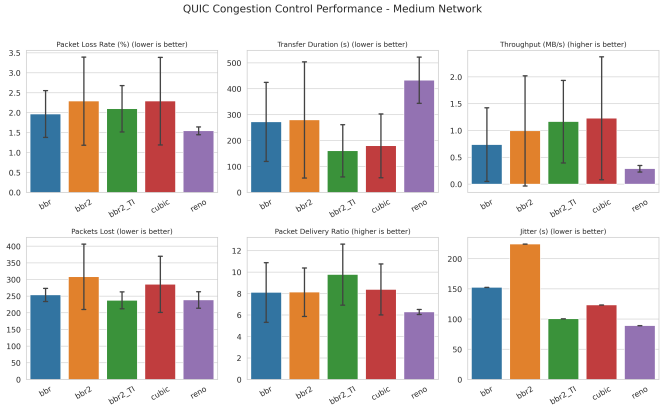

Fig. 4. Medium: 20 ms delay, 5 ms jitter, 1% loss; n=10 median, 95% CI.

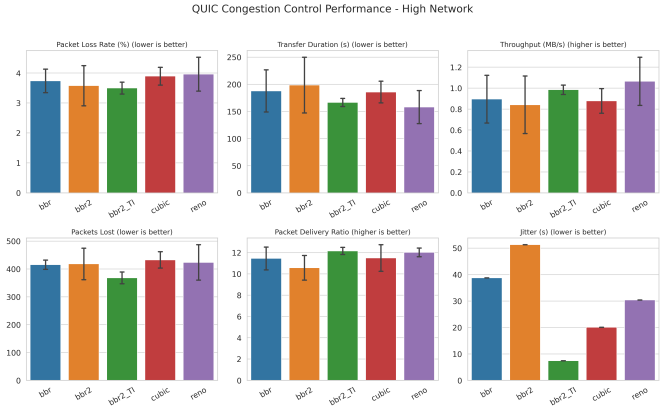

Fig. 5. High: 50 ms delay, 10 ms jitter, 2% loss; n=10 median, 95% CI.

VI. Conclusion

BBRv2-TI is presented as a utility-guided variant of BBRv2 designed for the strict latency, reliability, and stability demands of Tactile Internet systems. It employs an empirical gradient-free optimization framework that jointly minimizes RTT, jitter, and packet loss while maintaining throughput. By tuning eleven parameters within Cloudflare QUICHE, BBRv2-TI delivers lower jitter, reduced loss, and stable throughput under diverse network impairments. The framework runs entirely in user space, requires no kernel modification, and is lightweight enough for embedded or robotic platforms. Future work will focus on online utility adaptation and integration with intelligent scheduling to further enhance performance in next-generation TI networks.

Acknowledgment

This work was partially supported by the Palmer Department Chair Endowment.

References

[1] J. Sengupta, D. Dey, S. Ferlin, N. Ghosh, and V. Bajpai, "Accelerating tactile internet with quic: A security and privacy perspective," *arXiv preprint arXiv:2401.06657*, 2024.

[2] F. H. Fitzek, S.-C. Li, S. Speidel, T. Strufe, and P. Seeling, "Frontiers of transdisciplinary research in tactile internet with human-in-the-loop," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2021, pp. 1–6.

[3] M. H. Lashari, W. Batayneh, A. Khokhar, and S. Ahmed, "Enhanced position estimation in tactile internet-enabled remote robotic surgery using moesp-based kalman filter," *arXiv preFor instance, the reference template is not ieee reference template.print arXiv:2501.16485*, 2025.

[4] M. H. Lashari, S. Ahmed, W. Batayneh, and A. Khokhar, "A predictive approach for enhancing accuracy in remote robotic surgery using informer model," *Sensors*, vol. 25, no. 10, p. 3067, 2025.

[5] M. H. Lashari, W. Batayneh, and A. Khokhar, "Enhancing precision in tactile internet-enabled remote robotic surgery: Kalman filter approach," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2024, pp. 1189–1196.

[6] K. Antonakoglou, X. Xu, E. Steinbach, T. Mahmoodi, and M. Dohler, "Toward haptic communications over the 5g tactile internet," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3034–3059, 2018.

[7] F. Yang, Q. Wu, Z. Li, Y. Liu, G. Pau, and G. Xie, "Bbrv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing," *Computer Networks*, vol. 206, p. 108789, 2022.

[8] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review*, vol. 18, no. 4, pp. 314–329, 1988.

[9] R. Injong, "Cubic: A new tcpfriendly high-speed tcp variant," in *Proc. Third PFLDNet Workshop, Feb. 2005*, 2005.

[10] L. S. Brakmo and L. L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *IEEE Journal on selected Areas in communications*, vol. 13, no. 8, pp. 1465–1480, 2002.

[11] V. Arun and H. Balakrishnan, "Copa: Practical {Delay-Based} congestion control for the internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 329–342.

[12] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[13] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "{PCC} vivace:{Online-Learning} congestion control," in *15th USENIX symposium on networked systems design and implementation (NSDI 18)*, 2018, pp. 343–356.

[14] N. Rozen-Schiff, I. Pechtalt, A. Navon, and L. Bruckman, "Hercules: Heterogeneous requirements congestion control protocol," *arXiv preprint arXiv:2403.00590*, 2024.

[15] Z. Zhang, S. Li, Y. Ge, G. Xiong, Y. Zhang, and K. Xiong, "Pbq-enhanced quic: Quic with deep reinforcement learning congestion control mechanism," *Entropy*, vol. 25, no. 2, p. 294, 2023.