Insights into BBRv3's fairness over WebRTC

Matt Juntima (vqj9sq), Khuong Nguyen (cnr3jw), Sebastian Pop (qju9ta)

ABSTRACT

Recent congestion control research has focused on improving fairness and efficiency among elastic bulk flows, yet far less is known about how modern algorithms interact with latency-sensitive real-time applications. We study the coexistence behavior of BBRv3 with WebRTC traffic over legacy FIFO bottlenecks, a common deployment scenario where low-latency queue management is not available. Using a controlled experimental testbed, we evaluate a cloud-gaming-like WebRTC video stream competing against BBRv3 TCP cross-traffic under both lossless and lossy conditions. Our results show that BBRv3 preserves the operating throughput of WebRTC streams and avoids outright starvation, even when saturating the bottleneck. However, despite stable average bitrate and frame rate, BBRv3 induces significant latency inflation and delivery irregularity, leading to increased stall events and degraded Quality of Experience. These effects are amplified under modest packet loss, where WebRTC's congestion control reacts conservatively while BBRv3 continues aggressive probing. These findings demonstrate that improvements in throughput fairness do not necessarily translate into acceptable performance for real-time applications. We argue that coexistence evaluations must move beyond bandwidth-centric metrics and incorporate application-level Quality of Experience to accurately assess the impact of modern congestion control algorithms in mixed-traffic environments.

1 INTRODUCTION

Recent congestion control research has shifted from purely loss-based algorithms (e.g., TCP CUBIC) toward latency-aware and model-based designs. Instead of reacting only to packet loss, these algorithms explicitly estimate network properties and adjust sending behavior proactively. A representative example is Google's Bottleneck Bandwidth and Round-trip propagation time (BBR), which models the path using bottleneck bandwidth and minimum RTT, and paces packets near the bandwidth–delay product (BDP) while limiting in-flight data to avoid persistent queues [1]. In parallel, the Low Latency, Low Loss, Scalable Throughput (L4S) architecture proposes congestion controls such as TCP Prague that rely on fine-grained Explicit Congestion Notification (ECN) signals to reduce delay. These scalable ECN-based algorithms adjust their sending rates proportionally to the fraction of marked packets, enabling near-zero queuing delay when supported by DualQ Coupled Active Queue Management (AQM) at the bottleneck [2]. However, without such specialized queueing, scalable ECN flows are not safe for deployment over the general Internet and may suffer unfairness when competing with classic traffic [3].

When flows based on different congestion control paradigms share a bottleneck, fairness problems can occur. A well-known case is BBRv1 versus CUBIC: BBRv1 often starved CUBIC flows in shallow buffers by ignoring packet loss, while in deep buffers CUBIC could dominate by filling queues and forcing BBR to yield [4]. These issues motivated several revisions of BBR. BBRv3, released in 2023, explicitly targets improved coexistence with Reno and CUBIC. It introduces dynamic inflight limits and loss- and ECN-aware pacing adjustments, refining the ProbeBW behavior to reduce excessive bandwidth capture [5]. Recent evaluations indicate that BBRv3 achieves substantially better bandwidth sharing with CUBIC than earlier versions in many scenarios, especially in shallow-buffer environments where BBRv1 was highly aggressive [6]. Nevertheless, BBRv3 does not fully eliminate coexistence issues, and its designers acknowledge that challenges remain under heterogeneous traffic conditions.

An open question is how BBRv3 interacts with low-latency congestion control flows, particularly when L4S assumptions are violated. In principle, ECN-based flows such as TCP Prague should coexist with modern TCP algorithms if congestion signals are interpreted consistently. In practice, most Internet bottlenecks still deploy single FIFO queues with limited or no L4S support. Recent work by Sarpkaya *et al.* shows that under such legacy bottlenecks, Prague can experience severe throughput collapse when competing with BBRv3, with BBRv3 capturing the majority of the bandwidth [7]. This behavior is partly due to BBRv3's selective ECN response: ECN feedback is only used on short-delay paths, while on longer paths BBRv3 largely ignores ECN and behaves similarly to a loss-driven flow. As a result, an ECN-sensitive flow like Prague may back off aggressively while BBRv3 continues probing, leading to unfair outcomes.

Most prior fairness studies of BBR rely on elastic bulk traffic sources such as long-lived `iperf3` TCP flows. These workloads do not represent real-time applications, which have different traffic characteristics and performance objectives. Real-time media applications based on WebRTC, including video conferencing and cloud gaming, typically operate over UDP and use application-level congestion control such as Google Congestion Control (GCC). These flows are delay- and jitter-sensitive, and they aim to sustain stable frame delivery rather than maximize throughput [8, 9]. Prior measurements show that GCC-based WebRTC streams often yield bandwidth when competing with aggressive TCP flows, resulting in degraded video quality and stalls [10]. Google previously attempted to deploy BBR for WebRTC traffic, but this effort was reverted due to instability and poor real-time performance [11].

In this paper, we study the coexistence behavior of BBRv3 with real-time WebRTC traffic under realistic deployment conditions. We consider a scenario in which a BBRv3-enabled TCP flow competes with a WebRTC video stream over a shared bottleneck using standard FIFO queueing, representative of common access networks without L4S support. Unlike prior fairness studies that focus primarily on elastic bulk transfers, we evaluate performance from the perspective of real-time applications by measuring application-level Quality of Experience (QoE) metrics, including frame rate stability and freeze duration, in addition to throughput and latency. Specifically, we ask the following research question:
Can BBRv3 preserve Quality of Experience for WebRTC-based real-time applications when competing with bulk TCP traffic over legacy FIFO bottlenecks?

Our experimental results show that while BBRv3 preserves the operating throughput of WebRTC streams and avoids outright starvation, it induces substantial latency inflation and delivery irregularity that degrade user-perceived smoothness. These findings demonstrate that improvements in bandwidth fairness do not necessarily translate into safe coexistence for latency-sensitive applications, highlighting the need for additional safeguards such as queue management or application-aware adaptations when deploying modern congestion control algorithms in mixed-traffic environments.

The paper is organized as follows: Section 2 discusses the background and motivates the need for QoE-centric fairness evaluation. Section 3 describes our experimental methodology and the cloud gaming testbed implementation on FABRIC. Section 4 presents the evaluation results and analyzes the impact of BBRv3 on WebRTC performance along with discussions on issues we faced while implementing the framework. Section 5 reviews related work on congestion control and real-time applications. Section 6 outlines the limitations of our study. Finally, Section 7 concludes the paper. Our full implementation is available as open-source at: `https://github.com/knguyen2000/cozard`.

2 BACKGROUND AND MOTIVATION

Most prior work on congestion control fairness has focused on bandwidth allocation among elastic flows. Fairness is typically evaluated using throughput-based metrics, such as average rate or Jain's fairness index, measured under steady-state TCP traffic generated by tools like `iperf3` [12, 13]. While this methodology is suitable for comparing bulk data transfers, it does not capture the performance objectives of real-time applications, where user experience is primarily determined by delay variation, delivery regularity, and interruption events rather than sustained throughput.

Real-time media systems, particularly those based on WebRTC, operate under fundamentally different constraints. WebRTC applications employ adaptive encoders and application-layer congestion control to maintain smooth playback at a target frame rate [8]. Temporary reductions in available bandwidth can often be absorbed through quality adaptation, whereas short-term starvation or delay spikes can lead to visible freezes and playback stalls. As a result, two congestion control algorithms that appear fair under throughput-based metrics may produce significantly different QoE outcomes for real-time traffic [9, 10].

Despite this distinction, real-time traffic is rarely included as a first-class workload in congestion control fairness studies. Existing evaluations of BBR and its successors primarily consider interactions with loss-based TCP variants or ECN-capable L4S flows, using synthetic, elastic traffic patterns [4, 6]. These studies provide limited insight into how modern congestion control algorithms affect interactive applications that are both latency-sensitive and inelastic. In particular, it remains unclear whether improvements in bandwidth fairness translate into acceptable QoE for WebRTC when competing with aggressive background transfers.

This gap is especially relevant under current deployment conditions. Although the L4S architecture is designed to enable safe coexistence between low-latency and high-throughput traffic, its effectiveness depends on the presence of DualQ Coupled Active Queue Management at the bottleneck [2]. Such support is not yet widely deployed in access networks. Consequently, many real-world interactions between WebRTC and modern TCP variants such as BBRv3 occur over legacy FIFO bottlenecks, where congestion signals were not designed with real-time media in mind [7].

Motivated by these observations, this work shifts the focus of fairness evaluation from throughput-centric metrics to application-level impact. Rather than asking whether BBRv3 shares bandwidth evenly, we examine whether a WebRTC stream can maintain stable frame delivery when competing with BBRv3 under realistic network conditions. By measuring frame rate stability and freeze duration, we aim to characterize fairness from the perspective of real-time users rather than bulk traffic flows. This perspective is necessary to assess whether current congestion control designs are compatible with the increasing reliance on interactive, latency-sensitive applications.

3 METHODOLOGY

We implement a cloud gaming experiment testbed on the FABRIC platform to study the interaction between BBRv3

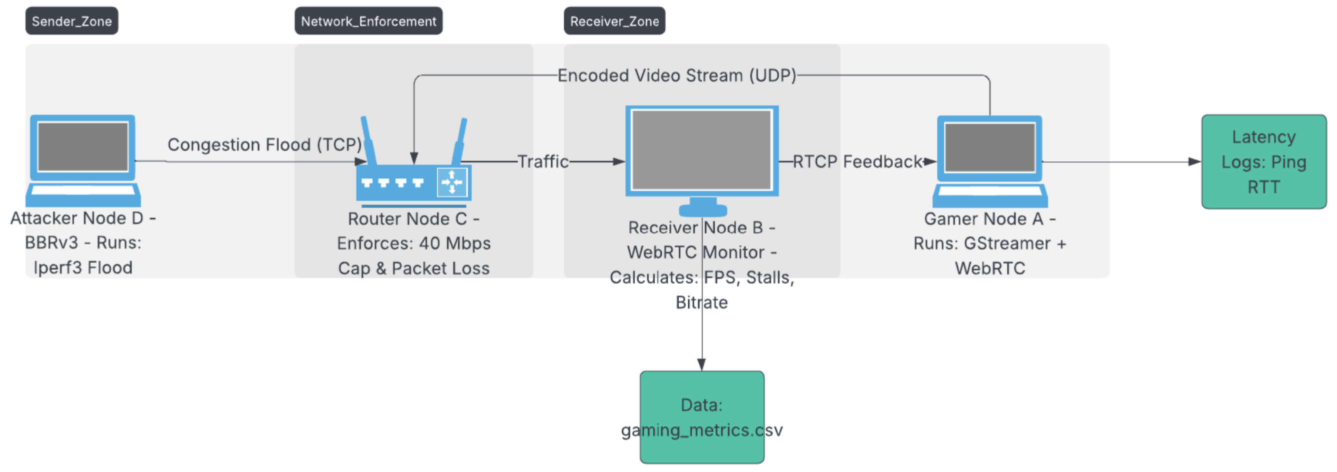
Sender_Zone
Network_Enforcement
Receiver_Zone
Encoded Video Stream (UDP)
Congestion Flood (TCP)
Traffic
RTCP Feedback
Attacker Node D - BBRv3 - Runs: Iperf3 Flood
Router Node C - Enforces: 40 Mbps Cap & Packet Loss
Receiver Node B - WebRTC Monitor - Calculates: FPS, Stalls, Bitrate
Gamer Node A - Runs: GStreamer + WebRTC
Latency Logs: Ping RTT
Data: gaming_metrics.csv

Figure 1: End-to-end experiment flow showing WebRTC video stream, RTCP feedback, competing BBRv3 TCP traffic, and measurement points for latency and QoE.

congestion control and WebRTC traffic under controlled and repeatable network conditions. The testbed provisions a dedicated network slice and executes experiments that combine real-time video streaming with competing TCP cross-traffic. The design ensures that observed performance effects are dominated by network behavior rather than host-side resource limitations.

All experiments are conducted within a single FABRIC site (SALT) to eliminate inter-site variability. The system consists of four physical nodes initially connected through a shared Layer 2 network, which is subsequently reconfigured into a logical Layer 3 topology to enforce a single shared bottleneck. Figure 1 provides a functional overview of the experiment, illustrating how WebRTC media traffic, RTCP feedback, and BBRv3-controlled TCP cross-traffic interact and where measurements are collected.

WebRTC video stream originates at Gamer A, which acts as the cloud gaming server. Gamer A streams a fixed high-motion video clip over UDP/WebRTC at a target rate of 60 frames per second. The video content is sourced from a publicly available high-motion test clip [14]. A constant frame rate is enforced to emulate interactive cloud gaming workloads, where timely frame delivery is more critical than throughput maximization. Using a fixed video source across all experiments ensures repeatability and avoids content-dependent variability in encoding behavior, allowing congestion effects to surface directly.

The video delivery pipeline on Gamer A is implemented using `GStreamer 1.0` and accessed via Python bindings (`gi`). To reflect production cloud gaming workloads, the pipeline prioritizes hardware acceleration. The video source is processed using the `nvh264dec` NVIDIA hardware decoder followed by `h264parse`, offloading compute-intensive decoding tasks to a Tesla T4 GPU. A fail-safe mechanism is implemented in the orchestration script: if the GPU pipeline fails to initialize (e.g., due to driver mismatches), the system automatically falls back to a software-based `uridecodebin` pipeline. Experimental runs are monitored to ensure that such fallback does not occur during measurements, as software decoding would introduce host-side latency unrelated to network behavior. While GPU acceleration is used for input video decoding, WebRTC transmission pipeline relies on software-based H.264 encoding. Specifically, decoded frames are passed to WebRTC stack implemented via `aiortc`, which performs real-time video encoding on the CPU prior to transmission. Hardware-accelerated encoding (e.g., `nvh264enc`) is not enabled in this implementation. As a result, GPU acceleration reduces the cost of frame generation, but the dominant encoding workload remains CPU-bound.

The transport layer is handled by `aiortc`, an asynchronous Python implementation of WebRTC standard. Frames are paced to target 60 FPS at 720p resolution with minimal buffering. Unlike video-on-demand services that rely on deep buffers, the sender pipeline prioritizes frame freshness to mimic the strict latency requirements of interactive cloud gaming. Receiver B acts as both WebRTC client and the performance monitor. It receives the video stream and generates RTCP feedback, which is sent back to Gamer A to form a closed-loop application-layer control path influencing encoder pacing and rate decisions. Competing background traffic is generated by Attacker D using `iperf3` with BBRv3 congestion control algorithm enabled. This TCP traffic aggressively probes for available bandwidth. Both WebRTC UDP stream and BBRv3 TCP flow traverse Router C.

Figure 1 also highlights the asymmetry between data and control paths. While WebRTC media packets flow from Gamer A to Receiver B, RTCP feedback travels in the reverse direction and may share the same bottleneck. As a result, increased queueing or packet loss at Router C can simultaneously delay video delivery and corrupt congestion feedback, amplifying its impact on real-time Quality of Experience.

Application-level QoE metrics, including frame rate, stall events, and bitrate, are computed at Receiver B and logged to persistent storage. In parallel, network-level latency measurements are collected using active probing (ICMP ping) to capture round-trip time variations induced by competing traffic. Separating application-level and network-level metrics allows us to distinguish throughput fairness from user-perceived performance degradation.

3.1 Experimental Setup

The physical topology comprises four nodes connected to a common Layer 2 network (`gaming_net`, 192.168.10.0/24):

- **Gamer A**: WebRTC sender representing the cloud gaming server.
- **Receiver B**: WebRTC receiver and application-level performance monitor.
- **Router C**: Intermediate node enforcing the network bottleneck.
- **Attacker D**: TCP traffic generator producing BBRv3-controlled cross-traffic.

Figure 2 shows the physical deployment of the four nodes within a single FABRIC site, interconnected through a shared Layer 2 network slice. This Layer 2 configuration is required by the FABRIC infrastructure to provide direct Ethernet connectivity between nodes and to allow full control over packet forwarding and traffic shaping behavior on intermediate nodes. Although all nodes are initially attached to the same Layer 2 broadcast domain, the experiment does not rely on Layer 2 forwarding semantics for traffic isolation. Instead, routing tables on each node are explicitly reconfigured to construct a linear Layer 3 topology in which all traffic between senders and receivers must traverse Router C:

Gamer A (10.2) ↔ Router C (10.1 / 20.1) ↔ Receiver B (20.2)

Attacker D injects BBRv3-controlled TCP traffic toward Receiver B through the same router, ensuring that WebRTC and TCP flows contend for identical queue and link resources.

This Layer 3 enforcement is essential for experimental validity. In a flat Layer 2 network, traffic may bypass the intended bottleneck due to implicit forwarding behavior or

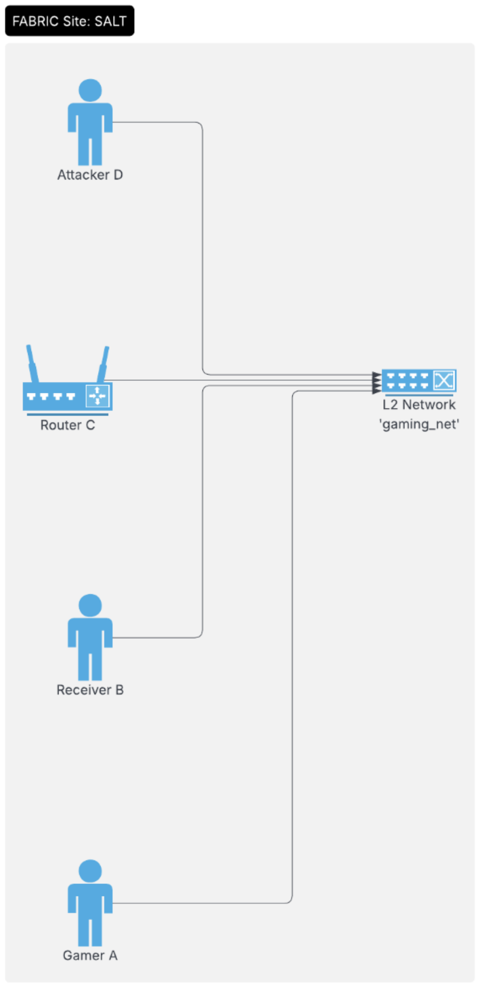

Figure 2: Physical deployment of the experiment on the FABRIC testbed at a single site, showing four nodes connected via a shared Layer 2 network.

testbed-specific optimizations. Explicit Layer 3 routing guarantees that Router C is the sole point of contention, localizing congestion effects and eliminating ambiguity about where queueing and loss occur. Figure 3 illustrates the resulting logical Layer 3 topology after routing reconfiguration. It highlights Router C as the enforced bottleneck and shows how both WebRTC traffic from Gamer A and BBRv3-controlled TCP traffic from Attacker D are forced to traverse the same queue and link toward Receiver B. This logical view makes explicit the shared contention point that is central to our fairness evaluation.

Router C enforces the bottleneck using Linux traffic control (`tc`). A Hierarchical Token Bucket (HTB) queue caps the link capacity at 40 Mbps, while a fixed queue size of 1000

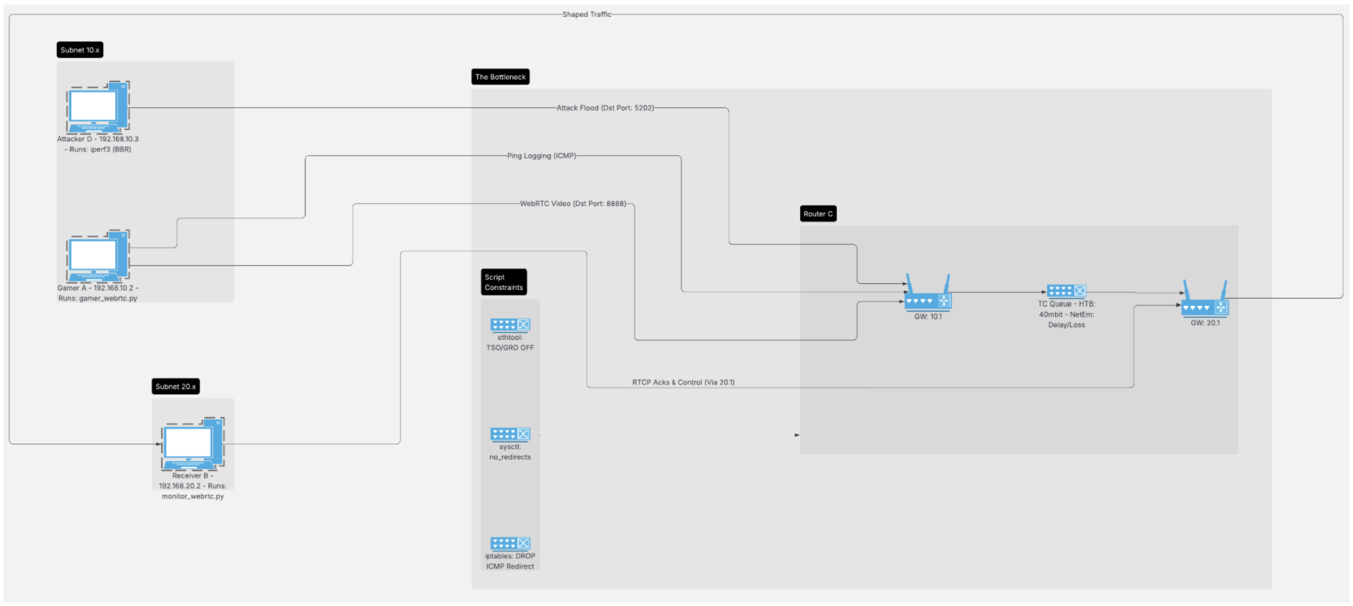
Shaped Traffic
Subnet 10.x
Attacker D - 192.168.10.3 - Runs: iperf3 (BBR)
Gamer A - 192.168.10.2 - Runs: gamer_webrtc.py
Subnet 20.x
Receiver B - 192.168.20.2 - Runs: monitor_webrtc.py
The Bottleneck
Attack Flood (Dst Port: 5202)
Ping Logging (ICMP)
WebRTC Video (Dst Port: 8888)
Script Constraints
ethtool: TSO/GRO OFF
sysctl: no_redirects
iptables: DROP ICMP Redirect
RTCP Acks & Control (Via 20.1)
Router C
GW: 10.1
TC Queue - HTB: 40mbit - NetEm: Delay/Loss
GW: 20.1

Figure 3: Logical Layer 3 topology after routing reconfiguration. Router C enforces bandwidth and queue constraints, forming a single shared bottleneck between WebRTC traffic and BBRv3 TCP cross-traffic.

packets provides a stable buffering regime. In selected experiments, controlled packet loss is introduced using `netem`. No L4S-specific queue management (e.g., DualQ or coupled AQM) is enabled, intentionally modeling legacy FIFO bottlenecks common in current Internet deployments. To ensure correctness of traffic shaping and measurement, network interface offloading features (TSO, GSO, GRO) are disabled on Router C. ICMP redirects are also disabled and filtered to prevent unintended route changes. These precautions ensure that observed packet delay, loss, and scheduling behavior accurately reflect the configured network conditions. Each experiment run follows a fixed, three-phase execution sequence designed to isolate specific protocol behaviors:

Baseline. WebRTC gaming stream runs in isolation for 5 seconds, establishing reference QoE metrics under uncongested conditions.

Wired Attack. BBRv3-controlled TCP cross-traffic is injected from Attacker D toward Receiver B through Router C, inducing sustained contention at the shared bottleneck while the gaming stream remains active.

Lossy Attack. The Wired Attack configuration is repeated with 2% random packet loss enabled at Router C using `netem`, allowing us to evaluate protocol behavior under non-congestive loss.

The cross-traffic remains active for the remainder of the experiment. Each run lasts 15 seconds in total, matching the duration of the video clip. All experiments are fully automated via a custom orchestration controller, which manages kernel configuration, verifies GPU availability, applies traffic control rules, and synchronizes data collection. Each configuration is repeated five times, and reported results are computed as averages across runs.

3.2 Evaluation Metrics

To quantify the interaction between BBRv3 and WebRTC, we define three primary categories of metrics: application-level QoE, transport-level fairness, and network latency.

Stall Events and Harm Factor. We interpret interactive performance primarily through frame delivery timing. A *stall event* is defined as any inter-frame arrival interval exceeding 200 ms. This threshold reflects the point at which video freezing becomes noticeable to human players. To compare performance across network conditions, we introduce the *Harm Factor* (H), defined as the ratio of total stall duration in a contention phase (D_{attack}) to the stall duration in the baseline phase ($D_{baseline}$):

$$H = \frac{D_{attack}}{D_{baseline}}$$

A Harm Factor of 1.0 indicates no degradation relative to the baseline, while values $H > 1.0$ quantify the multiplicative increase in freezing duration induced by competing traffic.

Throughput and Fairness. We measure the average throughput of WebRTC game stream (R_{game}) and BBRv3 cross-traffic (R_{attack}) over the 40-second steady-state period. To evaluate bandwidth sharing, we compute Jain's Fairness

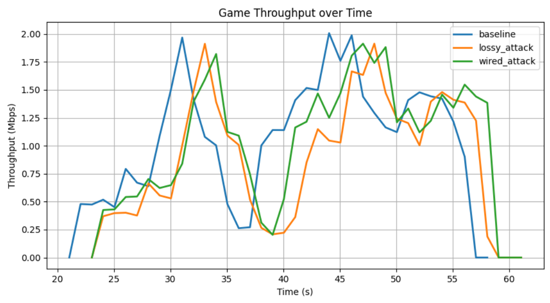

Figure 4: Time series of application-level goodput for WebRTC stream and competing BBRv3 TCP traffic under Baseline, Wired Attack, and Lossy Attack scenarios. WebRTC stream maintains a stable operating rate near 1 Mbps despite link saturation by BBRv3 cross-traffic.

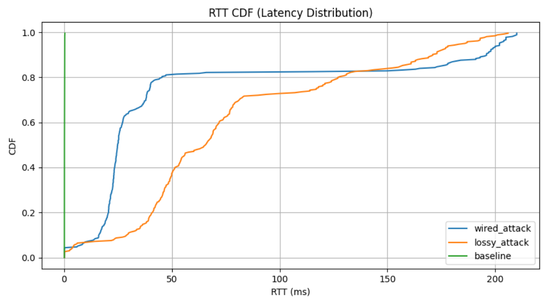

Figure 5: Cumulative distribution of RTT measured between WebRTC sender and receiver across scenarios. BBRv3 cross-traffic shifts RTT from a near-zero baseline to a broader distribution centered around 70–80 ms, with heavier tails under packet loss.

Index (J) between the two flows:

$$J = \frac{(R_{game} + R_{attack})^2}{2(R_{game}^2 + R_{attack}^2)}$$

J values range from 0.5 (absolute monopoly by one flow) to 1.0 (perfectly equal sharing). This metric allows us to assess whether BBRv3 acts as a "good neighbor" or aggressively starves the real-time flow.

Latency Distribution. Network latency is monitored via high-frequency ICMP probes (5 Hz) from the Gamer to the Receiver. We analyze the Cumulative Distribution Function (CDF) of Round-Trip Times (RTT) to characterize queueing delay dynamics independent of the application-layer buffering.

4 RESULTS AND DISCUSSION

This section presents an empirical evaluation of WebRTC and BBRv3 coexistence under controlled bottleneck conditions. We examine three scenarios as introduced in previous section: Baseline (uncongested), Wired Attack (40 Mbps BBRv3 cross-traffic), and Lossy Attack (40 Mbps BBRv3 with 1% random packet loss). Performance is evaluated using application-level QoE and network-level measurements. Table 1 summarizes key metrics averaged over the 40-second steady-state period.

Throughput Resilience Under BBRv3 Contention. The most prominent result is the *throughput stability* of WebRTC stream under BBRv3 cross-traffic. Figure 4 shows that WebRTC flow maintains a steady goodput of approximately 1 Mbps across all scenarios. During the Wired Attack, the competing BBRv3 flow saturates nearly the full link capacity (approximately 37–38 Mbps), yet the game stream sustains a throughput (1.01 Mbps) nearly identical to the uncongested baseline (1.04 Mbps). This indicates that BBRv3 does not starve low-rate real-time traffic, but instead allows WebRTC stream to retain its operating rate while probing for remaining bandwidth. This behavior contrasts with earlier observations of BBRv1 and loss-based TCP variants, which frequently suppressed low-rate UDP or real-time flows under saturation. The result confirms that BBRv3's revised pacing and inflight limits improve coexistence with heterogeneous traffic.

Latency Inflation. Although throughput is preserved, latency increases substantially under contention. Figure 5 and Figure 6 show that introducing BBRv3 cross-traffic shifts RTT from a near-zero baseline to a broad distribution centered around 70–80 ms, with excursions exceeding 150 ms in the lossy scenario. This latency inflation reflects queue buildup at the FIFO bottleneck and is consistent with BBR's model-based probing behavior. Importantly, RTT increases occur without a corresponding collapse in throughput, illustrating a key coexistence tradeoff: BBRv3 preserves bandwidth fairness but does not eliminate delay growth in legacy queues. From an interactive application perspective, RTT values below approximately 100 ms remain usable for casual cloud gaming, but the observed variability introduces frame arrival jitter that impacts smoothness.

Impact of Packet Loss on WebRTC Performance. The Lossy Attack scenario highlights the sensitivity of WebRTC's congestion control to non-congestive loss. With only 1% random packet loss, game throughput drops by approximately 16%, and average FPS degrades modestly (Figure 8). This behavior is not driven by BBRv3 directly, but by WebRTC's internal congestion control (GCC), which reacts conservatively to loss signals. While BBRv3 continues probing for bandwidth, WebRTC sender backs off to preserve delivery reliability, resulting in asymmetric adaptation. This confirms

Table 1: Summary of performance metrics across experimental scenarios.

Metric	Baseline	Wired Attack	Lossy Attack
Game Throughput (Goodput)	1.04 Mbps	1.01 Mbps	0.87 Mbps
Average RTT	0.16 ms	~78.4 ms	High variance
Video FPS	54.8	53.2	52.4
Total Stall Duration	2.59 s	3.10 s	3.84 s

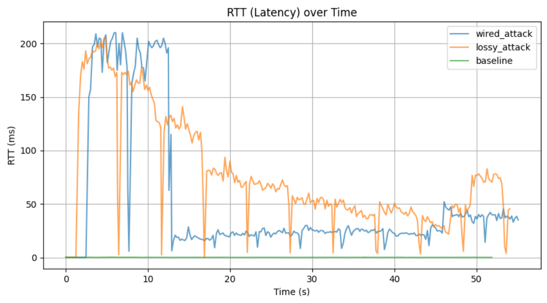

Figure 6: RTT time series showing queueing delay dynamics over the course of each experiment. Under BBRv3 cross-traffic, RTT exhibits sustained elevation and increased variability compared to the uncongested baseline.

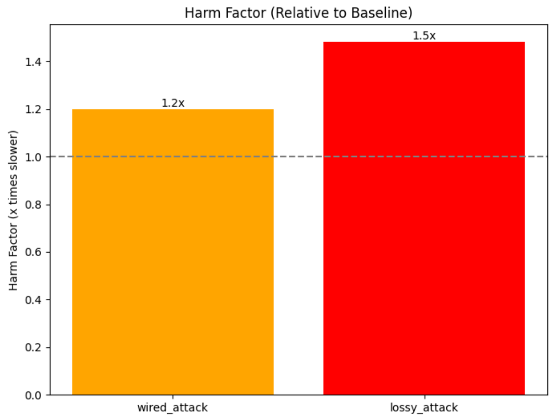

Figure 7: Harm Factor for stall duration, defined as the ratio of total stall time under attack relative to the baseline. Both Wired and Lossy Attack scenarios exhibit increased stall amplification, while absolute stall values are influenced by host-side processing artifacts.

that coexistence outcomes depend not only on TCP behavior, but also on the loss sensitivity of real-time congestion controllers.

Stall Events. Figure 7 reports the Harm Factor, defined as the ratio of total stall duration under attack relative to the baseline. Both attack scenarios exhibit increased stall time, with Harm Factors of 1.2× (Wired) and 1.5× (Lossy). However, an apparent anomaly emerges: the Baseline phase occasionally exhibits stall durations comparable to, or exceeding, those observed under attack. After multiple experiments and inspections, it appears that this phenomenon is not network-induced, but arises from a host-side processing artifact. The experiment relies on a 15-second high-motion video clip that is looped to span the full 40-second trial duration. Due to the unavailability of GPU-accelerated decoding in the test environment, decoding and seeking are performed in software on the CPU.

Each loop triggers a `seek()` operation that temporarily stalls frame delivery for approximately 500–800 ms. The precise duration of this stall varies due to OS scheduler noise, introducing run-to-run variability. As a result, baseline stall measurements include a deterministic, non-network noise floor that is independent of congestion conditions. Crucially, this artifact affects all scenarios equally and does not vary systematically with the presence of BBRv3 traffic. Attack phases introduce additional stall events on top of this baseline noise floor, as reflected by the Harm Factor, but absolute stall duration should be interpreted conservatively. Accordingly, stall metrics in this dataset are treated as supporting indicators of QoE degradation, while RTT inflation and throughput dynamics provide the primary evidence of network-induced coexistence effects. Figures 8 and 9 together illustrate why increased stall duration can coexist with stable average performance metrics. Although WebRTC stream maintains near-target throughput and frame rate under BBRv3 cross-traffic, QoE degrades due to increased delivery irregularity. This highlights a key limitation of average-based metrics: for real-time interactive applications, perceived smoothness is governed by short-term delay spikes and jitter rather than long-term rate allocation. As a result, throughput and average FPS alone are insufficient indicators of user experience under contention.

Figure 8 shows the instantaneous frame rate of WebRTC stream over time. Across all scenarios, the frame rate remains tightly clustered around the 60 FPS target, with only brief

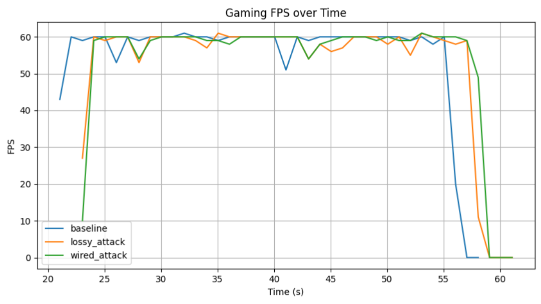

Figure 8: Instantaneous frame rate of WebRTC stream over time for all scenarios. Frame rate remains tightly clustered around the 60 FPS target with brief transient deviations and no sustained collapse under BBRv3 cross-traffic.

and shallow deviations. Neither the Wired nor the Lossy Attack induces a sustained FPS collapse, indicating that the application is not throughput-limited. The transient FPS dips correspond to short-lived delivery irregularities rather than persistent rate reduction. The sharp FPS drop near the end of each trace coincides with video termination and pipeline teardown and is not network-induced.

Importantly, we do not claim that stall amplification alone provides causal proof that BBRv3 induces video freezes. Due to the magnitude of the host-side decoding artifacts, the signal-to-noise ratio of absolute stall duration is insufficient to isolate fine-grained network-induced stall events. As a result, stall metrics are interpreted qualitatively and comparatively rather than as a standalone causal indicator. Our primary evidence of network-induced degradation instead derives from RTT inflation, increased delay variance, and delivery irregularity, which are not affected by host-side decoding behavior. The stall measurements serve only to illustrate how such network-level effects manifest perceptually when combined with real-time application constraints.

Fairness Perspective. Figure 9 summarizes bandwidth distribution under contention. The Jain's fairness index is low (0.53 in Wired Attack and 0.52 in Lossy Attack), indicating extreme inequality in rate allocation under the equal-entitlement assumption. This outcome is expected given the strong asymmetry between a greedy BBRv3 TCP flow that consumes nearly all available capacity and a WebRTC stream whose sending rate is constrained by application-level adaptation. For coexistence with real-time traffic, the more relevant question is whether WebRTC stream can sustain its operating rate in the presence of saturation cross-traffic. Despite BBRv3 claiming most excess bandwidth, WebRTC stream maintains approximately 1 Mbps in the Wired Attack scenario, indicating that it is not throughput-starved even under full bottleneck utilization. Nevertheless, the accompanying latency inflation and stall increases demonstrate that throughput preservation alone does not guarantee acceptable QoE.

5 RELATED WORK

Research on congestion control has traditionally evaluated fairness and performance using elastic, bulk data transfers. Benchmarking frameworks such as Pantheon [13] and fairness studies such as Schmitt *et al.* [12] primarily assess algorithms through steady-state throughput metrics, often summarized using Jain's fairness index. While this methodology is appropriate for comparing bulk TCP variants, it does not capture the performance objectives of latency-sensitive and inelastic applications.

BBR and Bandwidth Fairness. BBR [1] introduced a model-based approach to congestion control that departs from traditional loss-driven paradigms such as CUBIC [15]. Early experimental evaluations demonstrated that BBRv1 could achieve high throughput with low packet loss but exhibited severe coexistence problems. Hock *et al.* [4] showed that BBRv1 could either starve loss-based flows in shallow buffers or be dominated in deep-buffer scenarios, depending on queue configuration. Subsequent revisions culminated in BBRv3 [5], which explicitly targets improved coexistence through dynamic inflight limits and refined probing behavior. Recent evaluations report that BBRv3 achieves significantly better bandwidth sharing with CUBIC and Reno across a range of wired and wireless scenarios [6, 16]. However, these studies primarily rely on elastic TCP workloads and assess fairness almost exclusively through throughput-based metrics.

WebRTC Congestion Control and QoE. Real-time media applications based on WebRTC employ application-layer congestion control mechanisms such as Google Congestion Control (GCC) [9, 17]. Unlike bulk TCP flows, GCC prioritizes delivery regularity and low latency over throughput maximization. Prior studies show that WebRTC streams often yield bandwidth when competing with aggressive TCP flows, resulting in degraded video quality [10]. Attempts to adopt BBR for real-time media have been explored but ultimately abandoned. Iyengar and Thomson [11] document instability and poor latency behavior when BBR was applied to real-time traffic, leading Google to revert to GCC for WebRTC deployments. More recent work investigates BBR-based alternatives for WebRTC streaming [18], but these studies focus primarily on video quality adaptation rather than coexistence with competing TCP traffic.

Limitations of Throughput-Centric Fairness Metrics. Most existing fairness studies implicitly assume that equal bandwidth sharing is the primary objective. However, for

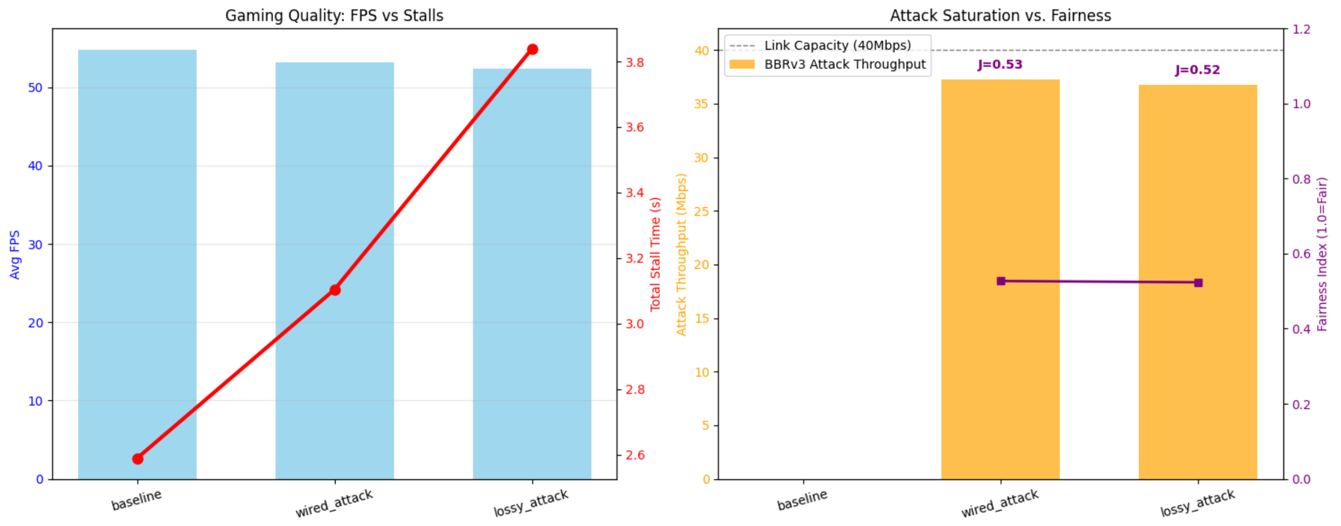

Figure 9: Summary of average performance and fairness metrics across scenarios, including frame rate, stall duration, and bandwidth allocation. While average throughput and FPS remain stable under BBRv3 cross-traffic, stall duration increases, revealing a disconnect between average metrics and perceived QoE.

real-time applications, short-term delay spikes and delivery irregularity can dominate user experience even when average throughput is preserved. As noted by Schmitt *et al.* [12], fairness metrics must be interpreted in the context of application objectives. Despite this observation, real-time traffic is rarely included as a first-class workload in congestion control evaluations. Existing studies of BBRv3 focus on throughput coexistence with other TCP variants or ECN-capable flows [6, 7], providing limited insight into application-level Quality of Experience (QoE) under mixed traffic conditions.

Positioning of This Work. In contrast to prior work, this paper evaluates BBRv3 coexistence from the perspective of a real-time WebRTC application operating over a legacy FIFO bottleneck. Rather than relying solely on throughput-based fairness metrics, we measure application-level QoE indicators, including frame rate stability and stall duration. By explicitly examining the disconnect between preserved throughput and degraded perceived smoothness, this study complements existing fairness evaluations and highlights limitations of bandwidth-centric metrics when applied to interactive, latency-sensitive workloads.

6 LIMITATIONS

Our experimental evaluation is subject to three primary limitations inherent to the testbed environment and scope.

Host-Side CPU Artifacts. First, the WebRTC sender relies on software-based H.264 encoding via `aiortc` and CPU-based video decoding/looping, as GPU acceleration was unavailable in the testbed. While production cloud gaming systems utilize hardware encoders (e.g., NVENC), our CPU-bound approach introduced deterministic processing artifacts. Specifically, the periodic `seek()` operation required to loop the 15-second reference clip generated consistent 500–800 ms stalls in frame delivery, independent of network conditions. These artifacts differ from network-induced jitter and occur deterministically across all scenarios. As a result, they introduce a constant noise floor that is mitigated, but not eliminated, by evaluating relative changes from the Baseline rather than absolute stall duration.

Infrastructure Constraints. Our experiments were conducted on the FABRIC testbed [19], which provided essential zero-cost access to programmable high-speed networking resources. However, reliance on this shared public infrastructure introduced specific operational constraints. First, the heterogeneity of the virtualized node environments contributed to the GPU driver and GStreamer library compatibility challenges that necessitated the software-based fallback described above. Second, platform resource policies: 24-hour slice lease limit and 4-hour credential rotation window—constrained the experimental design to short-term sessions, precluding multi-day longitudinal studies of BBRv3 behavior. Despite these logistical bounds, the testbed provided a reproducible and controlled environment sufficient for evaluating steady-state coexistence dynamics.

Synthetic Network Topology. Second, the network bottleneck was emulated using Linux Traffic Control (`tc-netem`) with a static FIFO queue and TBF. While this accurately models the queue dynamics of a capacity-constrained router, it does not capture the stochastic jitter, path re-routing, or AQM schemes found in commercial ISP networks. Consequently, the measured RTT values reflect a "clean-slate" bufferbloat scenario rather than the noisy latency profile of the public Internet.

Competition Scope. Finally, we evaluated a single-flow competition scenario (one gaming stream vs. one BBRv3 flow). In real-world environments, the gaming stream would likely compete with a diverse mix of TCP variants (Cubic, Reno) and background UDP traffic. Our focus on the "worst-case" 1-on-1 contention provides a lower bound on fairness performance but may not fully predict behavior in highly multiplexed residential broadband links. Future work should evaluate multi-flow and heterogeneous traffic mixes to assess how multiplexing affects these dynamics.

7 CONCLUSION

This paper examined the coexistence behavior of BBRv3 with real-time WebRTC traffic over legacy FIFO bottlenecks, focusing on application-level QoE rather than throughput-centric fairness metrics alone. Through controlled experiments on the FABRIC testbed, we evaluated a representative cloud-gaming-like WebRTC stream competing against aggressive BBRv3 TCP cross-traffic under both lossless and lossy conditions.

Our results show that BBRv3 avoids outright starvation of low-rate real-time flows. Even under full bottleneck saturation, the WebRTC stream consistently maintained its operating throughput, demonstrating a marked improvement over earlier BBR versions and many loss-based TCP algorithms. From a bandwidth allocation perspective, BBRv3 behaves as a “polite” competitor that claims excess capacity without suppressing the victim flow’s baseline demand. However, preserved throughput does not imply safe coexistence for real-time applications. Despite stable average bitrate and frame rate, BBRv3-induced queue buildup caused substantial latency inflation and delivery irregularity, leading to increased stall events and degraded perceived smoothness. These effects were exacerbated under even modest packet loss, where WebRTC’s congestion controller backed off conservatively while BBRv3 continued probing for bandwidth. This disconnect highlights a fundamental limitation of throughput-based fairness metrics when applied to latency-sensitive and inelastic workloads.

Taken together, our findings indicate that BBRv3’s fairness improvements primarily address bandwidth sharing among elastic flows and do not fully extend to application-level Quality of Experience for real-time media over legacy bottlenecks. As interactive applications such as cloud gaming and video conferencing continue to grow, coexistence must be evaluated not only in terms of rate preservation, but also in terms of delay stability and delivery regularity. Future work should explore mechanisms to bridge this gap, including queue management support (e.g., AQM or L4S-style signaling), sender-side adaptations that explicitly account for real-time traffic, and broader evaluations involving multiplexed and heterogeneous traffic mixes. Without such safeguards, modern congestion control algorithms may remain formally fair yet practically disruptive to latency-sensitive applications under common Internet deployment conditions.

REFERENCES

[1] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control. *ACM Queue*, 14(5):20–53, 2016.
[2] Bob Briscoe and Koen De Schepper. Low latency, low loss, scalable throughput (l4s) internet service. *IETF Internet-Draft*, 2020.
[3] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM ’10, pages 63–74. ACM, 2010.
[4] Markus Hock, Roland Bless, and Martina Zitterbart. An experimental evaluation of bbr congestion control. In *Proceedings of the IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2017.
[5] Neal Cardwell and Yuchung Cheng. Bbrv3: Improving fairness and coexistence of bbr. Technical report, Google, 2023. Linux TCP BBRv3 design document.
[6] Muhammad Ahsan, Muhammad Nabeel, Muhammad Mustansar, and Waqar Ashiq. Performance evaluation of bbrv3 with cubic and reno in a ubiquitous wired/wi-fi channel. *Journal of Computing and Biomedical Informatics*, 9(2):1–18, 2025.
[7] Fatih Berkay Sarpkaya, Ashutosh Srivastava, Fraida Fund, and Shivendra Panwar. To adopt or not to adopt l4s-compatible congestion control? understanding performance in a partial l4s deployment. In *Proceedings of the 26th International Conference on Passive and Active Measurement (PAM)*, pages 217–246. Springer, 2025.
[8] Harald Alvestrand. Webrtc overview: Real-time protocols for browser-based applications. Technical report, IETF, 2018.
[9] Stefan Holmer and Henrik Lundin. A google congestion control algorithm for real-time communication. In *Proceedings of the Packet Video Workshop*. IEEE, 2013.
[10] Luca De Cicco and Saverio Mascolo. Performance evaluation of webrtc congestion control. *Computer Communications*, 152:120–132, 2020.
[11] Janardhan Iyengar and Martin Thomson. Issues with using bbr for real-time media. Technical report, Google / IETF discussion, 2018.
[12] Paul Schmitt, Bob Briscoe, and Damon Wischik. Revisiting network fairness. In *Proceedings of the ACM SIGCOMM 2018 Conference*, pages 328–341. ACM, 2018.
[13] Francis Y. Yan, Junda Ma, Garrett Hill, Barath Raghavan, and Keith Winstein. Pantheon: The training ground for internet congestion-control research. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 731–743. USENIX Association, 2018.
[14] Google gtv video sample repository. http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/ForBiggerBlazes.mp4. Accessed 2025.
[15] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[16] Meng Yue, Kangchi Luo, Simin Yan, and Zhijun Wu. Research on bbr fairness in multiple scenarios. In *Proceedings of the 2025 4th International Conference on Big Data, Information and Computer Network*, pages 662–668. ACM, 2025.

[17] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*, pages 1–12. ACM, 2016.

[18] Rebecca Drucker, Aruna Balasubramanian, and Anshul Gandhi. Investigating webrtc bbr as an alternative to gcc for live video streaming. In *Proceedings of the 17th International Conference on Communication Systems & Networks (COMSNETS)*, pages 1–8. IEEE, 2025.

[19] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Monga, Kuang Ching Wang, Tom Lehman, Paul Ruth, and Ewa Deelman. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing*, 23(6):38–47, 2019.