Leveraging Cross-Directional Dependency in Realtime Interactive Streaming

Sen Lin
Northwestern University
Evanston, USA
sen.lin@u.northwestern.edu

Andre Chen
Northwestern University
Evanston, USA
andrechen2026@u.northwestern.edu

Kevin Zhikai Chen
Northwestern University
Evanston, USA
kevinchen2026@u.northwestern.edu

Aleksandar Kuzmanovic
Northwestern University
Evanston, USA
akuzma@northwestern.edu

Abstract

Realtime interactive streaming, an emerging paradigm that enables instantaneous two-way interactions with diverse streaming traffic, introduces new network challenges. This paper identifies a unique characteristic of this paradigm — cross-directional dependency. By leveraging this dependency, we propose a novel prioritization-based approach to optimize realtime interactive streaming. Our design aligns application-layer demands with transport-layer behavior, prioritizes critical traffic to mitigate buffer overflows caused by streaming microbursts, and transforms chaotic congestion into a predictable process. We extend the QUIC priority system to accommodate unreliable QUIC datagrams and develop an end-to-end framework for evaluating this emerging streaming paradigm. Extensive Internet-scale experiments validate the effectiveness of our system, demonstrating up to 9.4× reduction in motion-to-photon latency and 82× reduction in freeze frame rates.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Networks** → **Transport protocols**; • **Computer systems organization** → **Real-time systems**.

Keywords

Realtime streaming, Interactive media, Transport protocols, QUIC, Low latency, Traffic prioritization

ACM Reference Format:
Sen Lin, Andre Chen, Kevin Zhikai Chen, and Aleksandar Kuzmanovic. 2025. Leveraging Cross-Directional Dependency in Realtime Interactive Streaming. In *ACM Multimedia Asia (MMAsia '25), December 9–12, 2025, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3743093.3771029

1 Introduction

From video conferencing and cloud gaming to mixed reality (XR), realtime interactive streaming is revolutionizing digital experiences by enabling instantaneous, two-way interactions with ultra-low latency. This emerging streaming paradigm reshapes how users engage with content, transforming passive viewers into active participants. Recent years witnessed innovative applications, including holographic communication [1, 2, 5, 30, 40], digital twins [33, 43], turning sci-fi into reality. However, these applications require high-performance rather than consumer-grade networks, putting us one step away from their accessibility. In this paper, we will characterize realtime interactive streaming and present an approach tailored to its unique nature.

Unfortunately, the existing network ecosystem does not suit realtime interactive streaming. Figure 1 illustrates a holographic communication. It follows the status quo utilizing a Selective Forwarding Unit (SFU) [54] for exchanging data among participants.

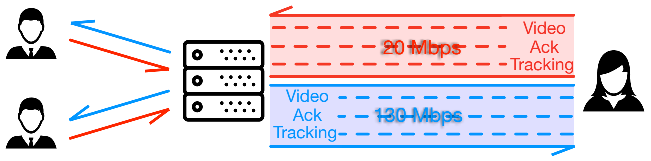

Figure 1: Illustration of network traffic patterns in realtime interactive streaming.

	5G [34]	Fixed Broadband [16]	Satellite [46]
Down	138-238 Mbps	467 Mbps	25-220 Mbps
Up	13-20 Mbps	71 Mbps	5-20 Mbps

Table 1: Network bandwidth asymmetry in the United States.

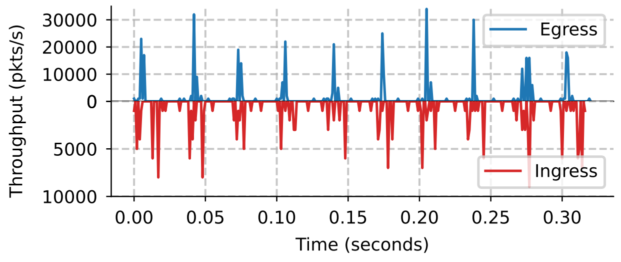

Figure 2: Microbursty throughputs in both directions of realtime interactive streaming.

First, compared to traditional audio/video (A/V) streaming, interactive streaming includes more diverse media types, such as sensor tracking. In addition, different traffic is often associated with distinct characteristics and demands. For example, sensor tracking is low-volume, high-frequency, latency sensitive but loss tolerant. This unique aspect prompts us to rethink multiplexing beyond simple A/V streaming.

Second, realtime interactive streaming involves heavy traffic in both directions. While Internet users now have ample bandwidth for bufferless 1080p or 4K streaming, this applies only to the downstream bandwidth, as upstream bandwidth is typically much lower. Recent measurements report such bandwidth asymmetry (Table 1) in the United States. Even in an ideal scenario, it is challenging for this upstream bandwidth to support 1080p or 4K live streaming according to the bitrate recommended by YouTube [20] not to mention 360°, virtual reality (VR), or 3D content.

Third, microbursts [6, 42] exacerbate this situation. Figure 2 shows the bidirectional throughputs in our testbed over a 0.5-second interval while streaming a 30 FPS video, with throughput calculated in 1 ms windows. Rather than being smoothly distributed over time, the throughput fluctuates significantly due to the completion of encoding new video frames, which causes the sender buffer to overfill with a significant amount of packets within a short time. This buffer behavior results in excessive packet drops and underutilization of egress bandwidth. Without lowering the encoding bitrate, counter measures [6, 23] trying to spread out heavy traffic, however, introduce latencies that hurt the realtime experience.

Last, state-of-the-art realtime streaming optimizations [3, 9, 13, 14, 18, 23, 29, 41, 44] focus on the single direction of the target stream but ignore the correlation of streams across directions, missing opportunities of further optimizations.

Our insights capitalize on the unique characteristic of cross-directional dependency in realtime interactive streaming. Cross-directional dependency is defined as streaming traffic in one direction depending on traffic in the other direction. Specifically, there are two types of cross-directional dependency. The first is cross-directional content dependency. For example, in holographic communication or VR streaming, a common practice is to stream rendered, perspective-projected, *view-dependent* content rather than raw 3D representations or full panoramic 360° videos [4, 8] to save bandwidth. To obtain view-dependent content, user-side tracking inputs, such as orientations or positions, are required. Consequently, obsolete user-uploaded tracking data can result in falsely processed content being downloaded for playback. The other is cross-directional transmission dependency. For instance, in the scenario depicted in Figure 1, when the upstream bandwidth is saturated, a high rate of upstream video retransmissions exacerbates congestion, leading to increased packet loss, including ACKs and tracking data. Excessive upstream ACK loss would not only cause increased downstream throughputs but also introduce playback delays or freezes.

Our proposal builds upon QUIC with the datagram extension [22, 36] by leveraging cross-directional dependency. While existing applications [8, 27] often use multiple connections for different traffic types, such a design leads to self-competition and delegates traffic management to the kernel, limiting application-level optimization. In contrast, QUIC's multiplexing and partial-reliability support allow us to consolidate diverse streams into a single connection with fine-grained bidirectional control. Additionally, partial reliability is essential for realtime streaming, as it allows senders to discard overdue data, e.g., expired video frames, enabling timely and application-aware delivery [35, 36, 39]. The key philosophy of our proposal is to prioritize critical traffic in both directions and turns chaotic congestion into a predictable one. To this end, we extend QUIC's priority system to support datagram prioritization and implement a comprehensive prioritization strategy across unreliable datagrams and reliable streams, aligning transmission behavior with application-level semantics and interaction goals.

	MsQuic [32]	**quinn** [38]	**quiche** [15]	**quic-go** [37]	**Chromium** [19]
Stream	✗	✓	✓	✗	✓
Datagram	✗	✗	✗	✗	✗

Table 2: Priority supports in the most popular QUIC libraries.

Our contributions are as follows: (1) We identify a cross-directional dependency in realtime interactive streaming and enable novel prioritization-based optimizations. (2) We extend the QUIC priority system to support datagram prioritization and propose a strategy tailored to streaming needs. (3) We develop RISE, an end-to-end emulation framework for realtime interactive streaming with diverse media types. (4) We demonstrate up to 9.4× and 82× reductions in motion-to-photon (MTP) latency and freeze frame rate in bidirectional view-dependent streaming. (5) We open source the extended QUIC library and the RISE framework to facilitate broader research on next-generation interactive streaming.[1]

2 Related Work

Prioritization Support. At the transport layer, both SCTP [47] and QUIC [22] support partial reliability and priority control, but their mechanisms differ. SCTP implements timed reliability, where a lifetime parameter can be attached to a stream to discard overdue (re)transmissions [39]. Additionally, the latest SCTP extensions include a priority-based scheduler under this context [48]. However, SCTP's communication model remains stream-based and lacks native datagram support, making it less suitable for realtime applications. As a result, WebRTC [7] adopts SCTP primarily for data channels rather than media channels. QUIC [22, 24], the foundation of HTTP/3 [12], approaches partial reliability differently. The datagram extension [36] introduces unreliable unordered datagrams as first-class citizens alongside reliable streams, significantly improving multiplexing flexibility for realtime scenarios. Although both QUIC streams and datagrams can theoretically be prioritized [22, 36], our survey of major QUIC libraries (Table 2) shows that only a few support prioritization for streams, and none support prioritization for datagrams. At the application layer, WebRTC exposes a four-level prioritization API, which can influence delivery via DSCP marking or proportional bandwidth allocation at the sender [7, 50]. However, these mechanisms lack fine-grained and preemptive control. In addition, DSCP marking is known to be fragile on the Internet [26].

[1]QUIC library: https://github.com/posoo/quinn-dg-priority; RISE framework: https://github.com/posoo/RISE-framework.

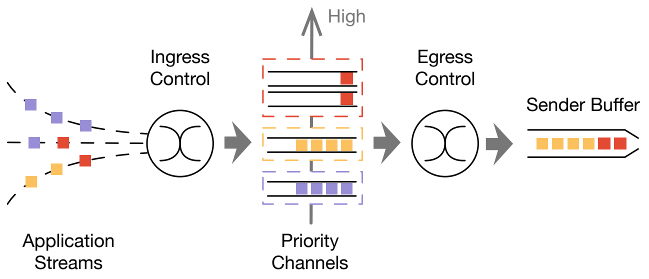

Figure 3: The proposed datagram processing pipeline with prioritization in QUIC.

Realtime Streaming Optimization. Forward Error Correction (FEC) has long been a key focus in realtime streaming optimization. Recent advances include adaptive redundancy schemes, such as FRACTaL [23] and Hairpin [29]; learning-based approaches, such as Tooth [9] and RL-AFEC [13]; and advanced coding designs, such as Tambur [41]. Additionally, bandwidth demands can be reduced through network-aware codecs, such as Salsify [18]; adaptive bitrate control, such as Mowgli [3]; and neural codecs, such as GRACE [14] and Gemino [44]. However, these techniques primarily target traditional one-way streaming, where the server pushes video content to a passive client. Recent efforts have begun exploring the realtime interactive streaming space. For example, LoopTailor [25] optimizes the VSync pipeline in cloud gaming. Our work complements these efforts by leveraging cross-directional dependency.

3 Design

3.1 Priority Control for QUIC Datagram

There is currently no established convention for implementing priority control for unreliable datagrams. In contrast, priority control for reliable streams is well-defined: High-priority stream data is more likely to be included in the next populated packet, while low-priority streams still get delivered eventually in a later stage. However, realtime applications demand unreliable transport to enable flexible retransmission at the application layer.

Our design focuses on establishing datagram priority control in QUIC to deal with streaming microbursts. We propose *priority channels*, which decouple unreliable application streams with specific priorities, enabling flexible and dynamic priority management. Ingress and egress controls work together to enforce priority control. Additionally, ingress control is responsible for capping the size of a *virtual* sender buffer to prevent bloating of the *actual* sender buffer and hence minimize latency. Egress control further helps mitigate starvation by ensuring fair scheduling among channels.

The pipeline of QUIC datagram transmission with priority control is illustrated in Figure 3, with the ingress and egress control procedures detailed in Algorithm 1. At the application layer, multiple unreliable streams may request transmitting datagrams with different priorities. A set of priority channels can be initialized either in advance or dynamically on demand. It is worth noting that an application stream may use different channels to send datagrams with varying priorities. For example, a video stream may prioritize key frame datagrams over predicted ones.

Algorithm 1: QUIC Datagram Priority Control

```
1  Procedure IngressControl($d, \hat{C}$):
2      $p \leftarrow$ priority($\hat{C}$) ;  // get the channel priority
3      $S \leftarrow \sum_{\forall C} |C|$ ; // get virtual sender buffer size
4      if $S < B$ then
5          $\hat{C} \leftarrow \hat{C} \cup \{d\}$
6      else
7          $C_{\min} \leftarrow \arg\min_{\forall C}$ priority($C$) | $C \neq \emptyset$;
8          if $p >$ priority($C_{min}$) then
9              dequeue($P_{\min}$);
10             $\hat{C} \leftarrow \hat{C} \cup \{d\}$;
11         else
12             drop($d$);
13 Procedure EgressControl($R$):
14     $p_{\max} \leftarrow \max\{$priority($C$) | $\forall C, |C| > 0\}$;
15     $\mathbb{C} \leftarrow \{C \mid \forall C,$ priority($C$) $= p_{\max}, |C| > 0\}$;
16     if $|\mathbb{C}| = 1$ then
17         $C' \leftarrow$ first($\mathbb{C}$);
18     else
19         $C' \leftarrow$ RR($\mathbb{C}$) ;        // round-robin selection
20     if size(first($C'$)) $\leq R$ then
21         return dequeue($C'$);
22     else
23         return null;
```

Ingress control is the first stage that determines whether to accept or drop an incoming datagram, and where to place it if accepted. We extend QUIC's datagram send API to include the associated priority channel ($\hat{C}$) for each datagram (d). After retrieving the channel priority (p) and the current size of the virtual sender buffer (S), if the buffer size does not exceed the buffer capacity (B), the datagram is immediately enqueued (L4–5). Otherwise, the datagram may still be accepted if its associated channel is not the one with the lowest priority. In that case, the lowest-priority channel will drop its oldest datagram, and the current datagram will be enqueued in the higher-priority channel (L6–10). If the current datagram belongs to the lowest-priority channel, it will be dropped without evicting any existing datagrams (L11–12).

Egress control is triggered when the underlying UDP socket becomes available. The highest-priority datagram that can be accommodated within the next populated QUIC packet is dequeued and moved to the actual sender buffer. Among channels with the same priority, datagrams are dequeued in a round-robin fashion to avoid starvation within equal-priority flows (L13–19). If the remaining space (R) is insufficient to fit the selected datagram, the current egress round is skipped, deferring transmission in anticipation of more available space in the next QUIC packet (L20–23).

While working locally at the sender side, our priority control mechanism not only proactively drops less important datagrams to protect critical ones, but also maintains an ordered priority queue to increase the likelihood of successful delivery of high-priority datagrams in subsequent networks. Importantly, we have no control over sender buffers in the kernel networking stack, network interfaces, or network middleboxes along the path, any of which can become congestion bottlenecks. While the optimal buffer size has long been a subject of debate, both academia and industry generally favor shallow sender buffers under modern Internet protocols and infrastructure [10, 11, 21, 28, 56]. As a result, allowing critical datagrams to enter the buffer earlier increases their chances of surviving subsequent congestion points.

3.2 Integration in Realtime Interactive Streaming

Extending QUIC with a datagram priority system enables cross-layer optimization by aligning transport-layer behaviors more closely with application-layer demands. It also extends QUIC's multiplexing concepts to cover QUIC datagrams. To illustrate the integration of this mechanism in realtime interactive streaming, we consider a holographic communication system based on bidirectional view-dependent streaming, as shown in Figure 1. In this system, each user continuously uploads tracking data (e.g., orientations), which is used by the communicating peer to render view-dependent video content, such as perspective-projected 360° video. Simultaneously, each user downloads video data and responds with acknowledgments (ACKs). The process is *symmetric* between interacting users, implying that each user simultaneously acts as both a *streamer* (encoding and sending video) and a *receiver* (sending tracking data and rendering received video). Our optimization goal is to minimize the MTP latency, which is the time between a tracking input and the corresponding video frame being displayed, and frame freezing.

Before diving into the analysis, we define the video encoding frequency as f_v, the motion sensor tracking frequency as f_t, and the round-trip time (RTT) between two interacting users as 2τ. In practice, $f_t \geq f_v$ is required to ensure smooth playback responsive to user movements. For the ease of explanation, we assume $f_t = 2f_v = \frac{1}{\tau}$ and zero processing delay. [2] In our proposed system, QUIC datagrams carry all three types of traffic, with tracking and video data treated as critical traffic.

Without datagram prioritization, the sender buffer behaves as a FIFO queue. Assuming the buffer has infinite capacity, the expected MTP latency is the sum of the RTT and the expected queuing delay for a tracking input (derivation omitted for brevity):

$$\mathbb{E}[L_{\text{MTP}}|\text{FIFO}] = 2\tau + \frac{R_v^2}{2B^2 f_v} \qquad (1)$$

where R_v denotes the video bitrate, B the available bandwidth, and f_v the video frame rate. For example, streaming a 30 FPS video with $R_v = \frac{2}{3}B$ and $\tau = 10$ ms yields an expected MTP latency of 27 ms. However, an infinite buffer is impractical in real-world scenarios, as it increases latency for all types of traffic. Additionally, due to the nature of video compression, frame sizes vary significantly, even under constant bitrate (CBR) encoding. For instance, a key frame can be up to two orders of magnitude larger than a predicted frame, increasing the burstiness. Furthermore, realtime streaming favors fresh data, and hence a bounded shallow buffer is preferred.

[2]For instance, video conferencing applications like Zoom [49] typically encode video at 30 frames per second (FPS). Devices such as the Meta Quest 2 headset have a tracking frequency around 60 Hz. The median RTT values in the United States are approximately 28 ms for cellular and 13 ms for fixed broadband users [45].

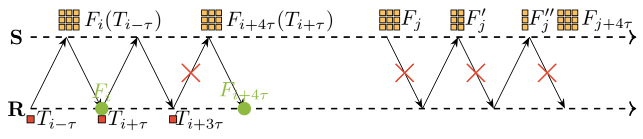

Figure 4: Realtime interactive streaming example. S and R stand for Streamer and Receiver respectively. Green dots indicate playback events.

Packet loss is a major culprit behind excessive MTP latency. As illustrated in Figure 4, at time $t = i$, the streamer generates and encodes a frame F_i based on tracking data received τ earlier. In this ideal case, at playback time $t = i+\tau$, the MTP latency is 2τ. Although the next tracking data ($T_{i+\tau}$) is successfully delivered, the subsequent tracking data ($T_{i+3\tau}$) is lost. As a consequence, frame $F_{i+4\tau}$ must use outdated tracking data ($T_{i+\tau}$), resulting in mismatched playback at $t = i + 5\tau$ and doubling the MTP latency to 4τ.

Similarly, failing to prioritize video datagrams can trigger a cascading effect that severely degrades streaming performance. Figure 4 depicts a scenario where the streamer encodes and transmits a frame F_j, which experiences partial loss during transmission. After one RTT, the streamer retransmits the lost data (F'_j), further increasing bandwidth usage and may exacerbating network congestion. The situation worsens if retransmitted data (F'_j) is also lost. Consequently, at time $t = j + 4\tau$, the streamer must simultaneously send both a new frame ($F_{j+4\tau}$) and retransmitted data (F''_j), intensifying network congestion. In addition, failure to decode frame F_j in a timely manner results in frame freezing. Furthermore, if the streamer decides to abandon frame F_j, then frame $F_{j+4\tau}$ must be encoded as a key frame to prevent error propagation, significantly increasing traffic consumption.

Based on the aforementioned insights, we assign the highest priority to tracking data, followed by video data, and the lowest to ACKs. Tracking is prioritized over video due to its extreme latency sensitivity and relatively small traffic volume. Reversing this order would risk starving tracking packets. Although ACKs are also low in volume, they are less latency-sensitive and can therefore be safely deprioritized. Counterintuitively, deprioritizing ACKs can actually improve streaming performance. Our prioritization orders datagrams by traffic type, taming congestion into a predictable process. Instead of being dropped randomly during buffer overflows, ACKs are dequeued right after bursty video traffic is drained, giving them a higher chance of successful delivery. While delayed ACKs may trigger occasional key frame insertions, they reduce the likelihood of frame freezing. Notably, this prioritization remains effective beyond the sender buffer, ensuring a higher success rate of delivery in downstream network queues (§3.1).

4 Evaluation

4.1 Evaluation Setup

Testbed. Realtime interactive streaming is an emerging paradigm, but lacks a standardized testbed for open research. Traditional applications like video conferencing and cloud gaming rarely involve symmetric or multi-stream traffic, while advanced scenarios such as holographic communication and digital twins are largely proprietary. To bridge this gap, we develop the **R**ealtime

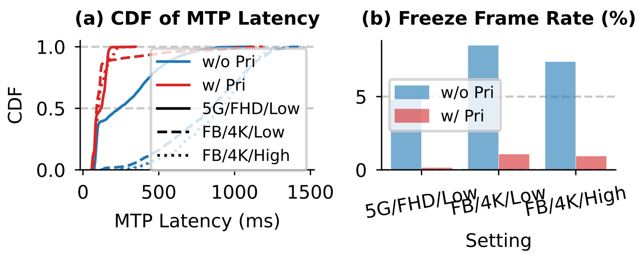

Figure 5: Transport-level results.

Interactive **S**treaming **E**mulation (RISE) framework, an end-to-end, high-performance Rust-based testbed supporting bidirectional streaming, parallel flows, view-dependent streaming, tracking replay, and multiple transport protocols. RISE builds on Ringmaster [31], which implements WebRTC streaming and has been widely adopted in prior work [17, 41, 55]. The QUIC extension is built on quinn [38].

Settings. We deploy RISE endpoints on two servers to emulate realtime interactive streaming between peers. Each endpoint equally acts as both a streamer and a receiver. For clarity, we refer to them as *Host* and *Guest*. Host running in a commercial data center (emulating a user behind an SFU) connects to Guest running on a campus network (simulating another user) via public Internet, where RTT is 21ms. We throttle the Guest's uplink bandwidth to 20 Mbps and 70 Mbps to simulate 5G/satellite (5G) and fixed broadband (FB) networks, as shown in Table 1. For input video, we use a 4K 360° VR gaming video from prior work [52]. Tracking data is synthesized at 60 Hz under three movement speeds (slow, moderate, fast). Each endpoint performs a perspective projection using the peer's latest 3DoF tracking with a 1-radian PoV. Video are encoded at 10, 30, and 40 Mbps for three settings: 5G/FHD/Low, FB/4K/Low, and FB/4K/High. Here, FHD and 4K denote 1080P and 2160P; Low and High indicate frame rates of 30 and 60 FPS. Each frame has a 1-second deadline. Bitrates are chosen based on YouTube's recommendations [20]. We primarily evaluate two transport schemes: our design with prioritization (w/ Pri) and baseline QUIC without prioritization (w/o Pri) under these three settings. Unless otherwise noted, results are reported from the Guest's perspective. We focus on QUIC to isolate the effect of datagram prioritization, as SCTP and WebRTC prioritization lack native datagram support and preemptive control, leaving comparative evaluation for future work.

4.2 Transport-Level Results

We first evaluate transport-level metrics by comparing our proposed approach against the baseline. By prioritizing tracking traffic, we achieve a median 2× reduction in MTP latency for streaming FHD/Low video over 5G networks. In fixed broadband networks, the median MTP latency is reduced by 9.4× for 4K/Low video and 8.8× for 4K/High video. Specifically, the 9.4× MTP latency improvement translates to a 730 ms reduction, allowing approximately 45 additional position updates at a tracking sensing frequency of 60 Hz. For the 95th and 99th percentile MTP latencies, our prioritization strategy yields reductions ranging from 2.3× to 5.7× and from 1.3× to 5.3×, respectively, across the three evaluated scenarios. This significantly lower MTP latency ensures more responsive graphical updates reflecting user movements.

Additionally, video traffic is prioritized to ensure smoother video playback. Playback smoothness is assessed by measuring freeze frame rates, following the definition in WebRTC's Statistics API [51]. When streaming FHD/Low video over 5G networks, we observe an average 82× reduction in freeze frame rate. For the other two scenarios over fixed broadband networks, the reductions exceed 8× on average. Across all tested conditions, prioritizing video traffic consistently maintains average freeze frame rates below 1%. These results confirm that assigning second-tier priority to video traffic effectively guarantees smoother video playback experience.

4.3 Content-Level Results

We have confirmed that our priority scheme significantly reduces MTP latency. We now examine how this improvement translates into enhanced quality-of-experience (QoE) by evaluating content-level metrics.

We first assess position mismatch, defined in 3DoF systems as the angular difference between user orientations at tracking and decoding time. As shown in Figure 6, our scheme achieves zero median mismatch under slow movement and reduces 95th and 99th percentile mismatches by 2.5 – 7.8× and 1.8 – 6.4×, respectively. For moderate and fast movement, gains increase further: median reductions reach 2.1 – 11× in 5G networks and 1.5 – 15× in fixed broadband networks; 95th and 99th percentile improvements range from 3 – 7× and 2.5 – 4× for moderate movement, and 2.5 – 28× and 4 – 18× for fast movement. These results underscore a butterfly effect, where small MTP latency differences can lead to amplified content-level discrepancies, even at low movement speeds.

We further evaluate per-frame similarity between decoded and reference videos, where the latter are rendered from ground-truth user positions at decoding time. We evaluate two similarity metrics: Structural Similarity Index Measure (SSIM) [53] and Peak Signal-to-Noise Ratio (PSNR). Notably, each video pair is temporally synchronized, ensuring that observed differences stem only from perspective projection effects. As shown in Figure 7, our scheme improves SSIM by 12%–68% in 5G networks and 12%–80% in fixed broadband networks, and improves PSNR by 10% – 80% and 37% – 93% in the two networks, respectively. However, the improvements diminish at higher movement speeds. This reflects a fundamental limitation of SSIM and PSNR: they are designed to quantify quality degradation of identical content, not perceptual differences arising from changes in perspective. We therefore view position mismatch as a more appropriate QoE metric—but all three metrics validate the effectiveness of prioritization on user experience.

4.4 Ablation Study

To gain a deeper understanding of how datagram prioritization leverages cross-directional dependency to optimize realtime interactive streaming, we conduct an ablation study to analyze the impact of prioritization directions and the effective range of prioritization. In this study, we focus on the streaming setting of FB/4K/Low.

Directional Impacts. In addition to duplex prioritization (w/ Pri) and the baseline (w/o Pri), we evaluate two asymmetric schemes:

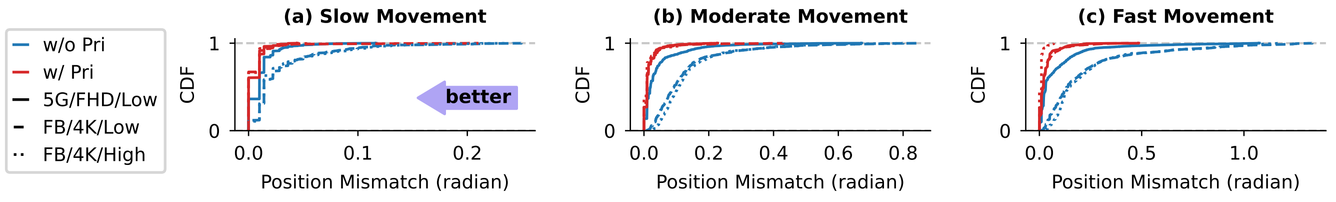

Figure 6: CDFs of position mismatch under different movement speeds.

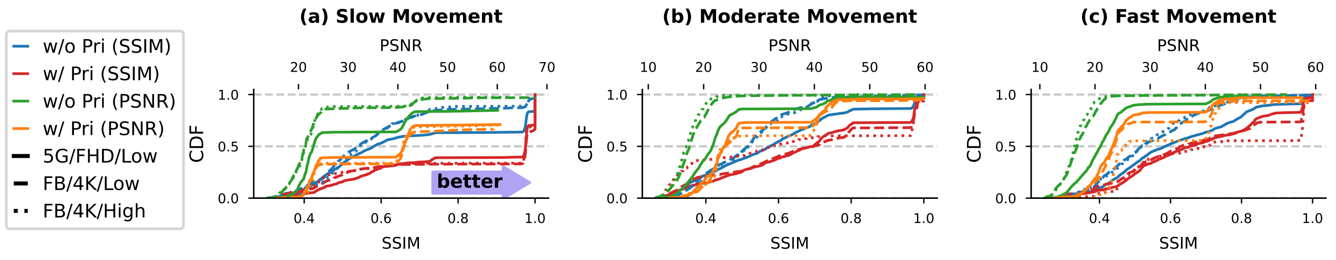

Figure 7: CDFs of similarity scores under different movement speeds.

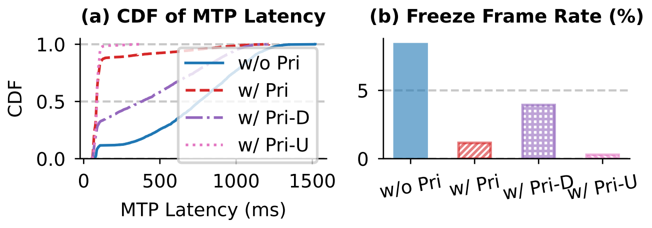

Figure 8: Transport-level results under different prioritization directions. “Pri-D”, “Pri-U”, and “Pri” denote enabling prioritization on downstream, upstream, and both directions.

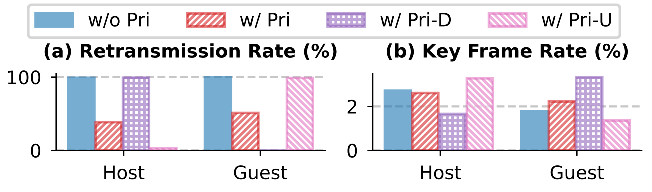

Figure 9: Retransmission rates and key frame rates under different prioritization directions.

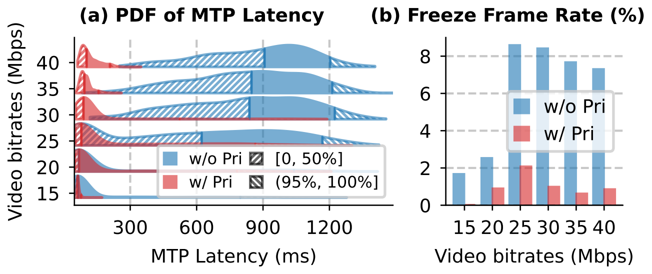

Figure 10: Transport-level results under different video encoding bitrates.

upstream-only (w/ Pri-U) and downstream-only (w/ Pri-D) datagram prioritization.[3] As shown in Figure 8, enabling prioritization solely on the upstream, the bottleneck direction, yields the most significant gains, reducing median MTP latency and average freeze frame rate by 9× and 32×, respectively. In contrast, downstream-only prioritization achieves 2× and 2.4× reductions. These results confirm that microbursting issues primarily occur in the constrained upstream. Interestingly, downstream prioritization still helps despite the downstream link's ample bandwidth (1 Gbps vs. 30 Mbps demand), indicating that burst-induced buffer overflows can still occur under low bandwidth pressure. However, the mechanisms of improvement differ between upstream and downstream prioritization. To understand this distinction, Figure 9 presents retransmission and key frame rates under all schemes.

In the downstream-only case, Host's retransmission rate remains similar to the baseline, as retransmission signals (ACKs) from Guest are neither prioritized nor deprioritized. However, Host encodes only half as many key frames, since its video datagrams are prioritized over ACKs and thus delivered more promptly. This also contributes to reduced MTP latency at the Guest. Meanwhile, Guest experiences almost no retransmissions but shows elevated key frame rates. This is because ACKs from Host are deprioritized, leading to frequent loss or delay of retransmission feedback and forcing Guest to encode more key frames. As a result, burstiness on the bandwidth-limited Guest-to-Host link worsens, partially offsetting the gains in MTP latency.

In the upstream-only case, Guest's MTP latency drops directly due to the prioritization of its latency-sensitive tracking data. Freeze frame rates also decrease, as Guest's ACKs are mostly delayed rather than dropped (§3.2), This is evidenced by Host's near-zero retransmission rate and similar key frame rate. Although Host's key frame rate slightly increases, Guest's ample downstream bandwidth absorbs the additional traffic from Host.

[3]w/ Pri-U and w/ Pri-D enable prioritization only on the Guest and Host, respectively. These asymmetric schemes are unfair in symmetric setups, evaluated only for ablation.

Effective Range. In fixed broadband networks (70 Mbps uplink), we vary video bitrates from 15 to 40 Mbps to evaluate the effective range of datagram prioritization. As shown in Figure 10, reductions in MTP latency grow with the bandwidth surplus. Above 25 Mbps, prioritization reduces median MTP latency by 7–9×, 95th-percentile latency by 1.3–10×, and freeze frame rates by 4–12×. Across all bitrates, it consistently keeps median MTP latency below 103 ms and freeze rates under 2%, demonstrating robustness. Even at 15 Mbps, it reduces 95th-percentile MTP latency by 10×, mitigating microbursting under light loads.

5 Conclusion

In this paper, we identify the cross-directional dependency unique to realtime interactive streaming. This insight motivates new optimization directions: prioritizing critical traffic in both directions to mitigate microbursting without added latency, and turning chaotic congestion into a predictable process. We extend QUIC's priority system to support datagram prioritization, enhancing its multiplexing capability. To evaluate our design, we develop RISE, an end-to-end testbed tailored for realtime interactive streaming with diverse bidirectional traffic patterns. Extensive Internet-scale experiments show up to 9.4× MTP latency reduction and 82× lower freeze rates. Even under low bandwidth pressure, our prioritization strategy consistently outperforms vanilla QUIC. Future work will explore integrating FEC, adaptive priority, and middlebox coordination to further exploit cross-directional dependencies for more demanding applications.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation under Grants 2310927 and 2226107.

References

[1] 2025. Evercoast - 4D Spatial Volumetric Studio. Retrieved March 1, 2025 from https://evercoast.com
[2] 2025. Proto Hologram - Holographic Communications Platform. Retrieved March 1, 2025 from https://protohologram.com
[3] Neil Agarwal, Rui Pan, Francis Y Yan, and Ravi Netravali. 2025. Mowgli: Passively Learned Rate Control for Real-Time Video. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 579–594.
[4] Volga Aksoy, Irad Ratmansky, and Amanda Watson. 2019. How does Oculus Link Work? The Architecture, Pipeline and AADT Explained. Retrieved March 1, 2025 from https://developers.meta.com/horizon/blog/how-does-oculus-link-work-the-architecture-pipeline-and-aadt-explained/
[5] Akihito Akutsu, Akira Ono, Hideaki Takada, Yoshihide Tonomura, and Maiko Imoto. 2016. 2020 Public Viewing–Kirari! Immersive Telepresence Technology. *NTT Technical Review* 14, 12 (Dec. 2016), 30–35.
[6] Mark Allman and Ethan Blanton. 2005. Notes on burst mitigation for transport protocols. *ACM SIGCOMM Computer Communication Review* 35, 2 (2005), 53–60.
[7] Harald T. Alvestrand. 2021. Transports for WebRTC. RFC 8835. doi:10.17487/RFC8835
[8] alvr org. 2025. ALVR - Air Light VR. Retrieved March 1, 2025 from https://github.com/alvr-org/ALVR
[9] Congkai An, Huanhuan Zhang, Shibo Wang, Jingyang Kang, Anfu Zhou, Liang Liu, Huadong Ma, Zili Meng, Delei Ma, Yusheng Dong, et al. 2025. Tooth: Toward Optimal Balance of Video QoE and Redundancy Cost by Fine-Grained FEC in Cloud Gaming Streaming. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 635–651.
[10] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing router buffers. *ACM SIGCOMM Computer Communication Review* 34, 4 (2004), 281–292.
[11] Wei Bai, Shuihai Hu, Kai Chen, Kun Tan, and Yongqiang Xiong. 2020. One more config is enough: Saving (DC) TCP for high-speed extremely shallow-buffered datacenters. *IEEE/ACM Transactions on Networking* 29, 2 (2020), 489–502.
[12] Mike Bishop. 2022. HTTP/3. RFC 9114. doi:10.17487/RFC9114
[13] Ke Chen, Han Wang, Shuwen Fang, Xiaotian Li, Minghao Ye, and H Jonathan Chao. 2022. RL-AFEC: adaptive forward error correction for real-time video communication based on reinforcement learning. In *Proceedings of the 13th ACM Multimedia Systems Conference*. 96–108.
[14] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. GRACE: Loss-Resilient Real-Time Video through Neural Codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 509–531.
[15] Cloudflare. 2025. quiche. Retrieved March 1, 2025 from https://github.com/cloudflare/quiche
[16] Federal Communications Commission. 2024. Measuring Fixed Broadband - Thirteenth Report. Retrieved March 1, 2025 from https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-fixed-broadband-thirteenth-report
[17] Hao Fang, Haoyuan Zhao, Feng Wang, Yi Ching Chou, Long Chen, Jianxin Shi, and Jiangchuan Liu. 2024. Streaming Media over LEO Satellite Networking: A Measurement-Based Analysis and Optimization. *ACM Transactions on Multimedia Computing, Communications and Applications* (2024).
[18] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 267–282.
[19] Google. 2025. QUIC in The Chromium Projects. Retrieved March 1, 2025 from https://www.chromium.org/quic/playing-with-quic/
[20] Youtube Help. 2025. Choose live encoder settings, bitrates, and resolutions. Retrieved March 1, 2025 from https://support.google.com/youtube/answer/2853702?hl=en
[21] Geoff Huston. 2019. Sizing the buffer. *APNIC Blog* (2019). Retrieved March 1, 2025 from https://blog.apnic.net/2019/12/12/sizing-the-buffer/
[22] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. doi:10.17487/RFC9000
[23] Balázs Kreith, Varun Singh, and Jörg Ott. 2017. FRACTaL: FEC-based rate control for RTP. In *Proceedings of the 25th ACM international conference on Multimedia*. 1363–1371.
[24] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.
[25] Yang Li, Jiaxing Qiu, Hongyi Wang, Zhenhua Li, Feng Qian, Jing Yang, Hao Lin, Yunhao Liu, Bo Xiao, Xiaokang Qin, et al. 2025. Dissecting and Streamlining the Interactive Loop of Mobile Cloud Gaming. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 595–611.
[26] Sen Lin, Jianfeng Wang, and Aleksandar Kuzmanovic. 2024. Optimizing Traffic in Public-Facing Data Centers Amid Internet Protocols. In *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*. IEEE, 1–12.
[27] LizardByte. 2025. Sunshine - Self-hosted game stream host for Moonlight. Retrieved March 1, 2025 from https://github.com/LizardByte/Sunshine
[28] Nick McKeown, Guido Appenzeller, and Isaac Keslassy. 2019. Sizing router buffers (redux). *ACM SIGCOMM Computer Communication Review* 49, 5 (2019), 69–74.
[29] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 907–926.
[30] Meta. 2025. Codec Avatars. Retrieved March 1, 2025 from https://about.meta.com/realitylabs/codecavatars/
[31] Microsoft. 2024. RingMaster. Retrieved March 1, 2025 from https://github.com/microsoft/ringmaste
[32] Microsoft. 2025. MsQuic. Retrieved March 1, 2025 from https://github.com/microsoft/msquic
[33] Nvidia. 2025. Omniverse Digital Twin. Retrieved March 1, 2025 from https://docs.omniverse.nvidia.com/digital-twins/latest/index.html
[34] Opensignal. 2025. Mobile Network Experience Report. Retrieved March 1, 2025 from https://www.opensignal.com/reports/2025/01/usa/mobile-network-experience
[35] Mirko Palmer, Malte Appel, Kevin Spiteri, Balakrishnan Chandrasekaran, Anja Feldmann, and Ramesh K Sitaraman. 2021. VOXEL: Cross-layer optimization for video streaming with imperfect transmission. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 359–374.
[36] Tommy Pauly, Eric Kinnear, and David Schinazi. 2022. An Unreliable Datagram Extension to QUIC. RFC 9221. doi:10.17487/RFC9221
[37] quic go. 2025. quic-go. Retrieved March 1, 2025 from https://github.com/quic-go/quic-go
[38] quinn org. 2025. quinn. Retrieved March 1, 2025 from https://github.com/quinn-rs/quinn
[39] Michael A. Ramalho, Qiaobing Xie, Randall R. Stewart, Michael Tüxen, and Phillip Conrad. 2004. Stream Control Transmission Protocol (SCTP) Partial Reliability

Extension. RFC 3758. doi:10.17487/RFC3758

[40] Microsoft Research. 2025. Holoportation. Retrieved March 1, 2025 from https://www.microsoft.com/en-us/research/project/holoportation-3/

[41] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[42] Taveesh Sharma, Tarun Mangla, Arpit Gupta, Junchen Jiang, and Nick Feamster. 2023. Estimating webrtc video qoe metrics without using application headers. In *Proceedings of the 2023 ACM on Internet Measurement Conference*. 485–500.

[43] Siemens. 2025. Digital Twin. Retrieved March 1, 2025 from https://www.siemens.com/global/en/products/automation/topic-areas/digital-enterprise/digital-twin.html

[44] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. 2024. Gemino: Practical and robust neural compression for video conferencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 569–590.

[45] Speedtest. 2025. United States Median Country Speeds. Retrieved March 1, 2025 from https://www.speedtest.net/global-index/united-states

[46] Starlink. 2025. Specification. Retrieved March 1, 2025 from https://www.starlink.com/legal/documents/DOC-1400-28829-70

[47] Randall R. Stewart, Michael Tüxen, and karen Nielsen. 2022. Stream Control Transmission Protocol. RFC 9260. doi:10.17487/RFC9260

[48] Randall R. Stewart, Michael Tüxen, Salvatore Loreto, and Robin Seggelmann. 2017. Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol. RFC 8260. doi:10.17487/RFC8260

[49] Zoom Support. 2024. Accessing meeting and phone statistics. Retrieved March 1, 2025 from https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0070504

[50] W3C. 2021. WebRTC Priority Control API. Retrieved March 1, 2025 from https://www.w3.org/TR/webrtc-priority/

[51] W3C. 2025. Identifiers for WebRTC's Statistics API. Retrieved March 1, 2025 from https://www.w3.org/TR/webrtc-stats/

[52] Shibo Wang, Shusen Yang, Hailiang Li, Xiaodan Zhang, Chen Zhou, Chenren Xu, Feng Qian, Nanbin Wang, and Zongben Xu. 2022. SalientVR: Saliency-driven mobile 360-degree video streaming with gaze information. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 542–555.

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[54] Magnus Westerlund and Stephan Wenger. 2015. RTP Topologies. RFC 7667. doi:10.17487/RFC7667

[55] Kichang Yang, Minkyung Jeong, Juheon Yi, Jingyu Lee, KyoungSoo Park, and Youngki Lee. 2024. Logan: Loss-tolerant Live Video Analytics System. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1314–1329.

[56] Sharada Yeluri. 2023. Sizing Router Buffers - Small is the New Big. *Juniper TechPost* (2023). Retrieved March 1, 2025 from https://community.juniper.net/blogs/sharada-yeluri/2023/02/22/sizing-router-buffers