QCON: Seamless QoE-Aware 5G Streaming via Multi-Connectivity

Goodsol Lee[†] Junhong Min[‡] Seyeon Kim[§] Juheon Yi[¶]
Kwang Taik Kim[‖] Mung Chiang[‖] Sangtae Ha[‡] Kyunghan Lee[†] Saewoong Bahk[†]
[†]Seoul National University [‡]University of Colorado Boulder [§]Korea University
[¶]Microsoft Research [‖]Purdue University

Abstract

Mobile real-time video streaming (RTS) applications—cloud gaming and AR/VR—require consistent high throughput and low latency to satisfy user Quality of Experience (QoE), yet today's wireless links fluctuate wildly. While multi-path solutions seem promising to tackle such single-link fluctuations, existing transport-level solutions require multiple cellular subscriptions, which most users don't have. In this paper, we leverage 5G multi-connectivity, which allows simultaneous connection to multiple base stations (e.g., 5G and 4G) and is already deployed in commercial networks. However, our measurements show RTS applications still suffer from single-link fluctuations due to operators' deliberate policies restricting multi-connectivity to conserve 4G backup links regardless of application demands. To optimize application QoE while respecting operator policies, we present QCON, a QoE-driven multi-connectivity solution that efficiently utilizes backup links based on precise application QoE. For practical deployment, we design QoE Monitor to infer application QoE within the RAN and develop multi-link scheduling to optimize both QoE and radio resource efficiency. We also design priority-based re-injection utilizing RAN link recovery mechanism to prevent video stalls. Our prototype implementation of QCON on a RAN intelligent controller within an Open-RAN testbed demonstrates 2.1× improvements of bitrates, enhancing tail frame rates by 4-5× with efficient backup link use compared to existing multi-link scheduling schemes.

1 Introduction

Real-time video streaming (RTS) has become the backbone for a wide range of interactive and immersive applications (apps), including video conferencing [1, 6], cloud gaming [60, 63], and virtual/augmented reality [24, 85]. These services increasingly cater to mobile users, from passengers in autonomous vehicles to travelers on public transit, placing stringent performance requirements on today's wireless networks. However, the requirements for RTS are particularly high to provide a good QoE (Quality of Experience) for users: a single 1080p gaming stream at 60 FPS consistently requires 25 Mbps [63] of throughput and per-frame latencies under 100 ms [58]. While 5G radio access networks (RAN) are designed to meet these goals, real-world deployments still exhibit frequent disruptions—especially in mobile scenarios [36, 43]. Even brief packet losses or latency spikes can trigger cascading stalls, significantly degrading the user experience of apps [26, 88].

A well-known approach to mitigating link disruptions is the multi-path transport solution [26, 47, 76, 88]. These techniques can send packets across multiple links to maintain continuous traffic flow, even if one link fails. While promising, they face two significant limitations: (1) Practical deployment remains challenging, as mobile users must subscribe to multiple cellular plans from different operators. This presents a significant adoption barrier, with only 0.5% of US mobile users utilizing multi-carrier plans [65], which typically cost more than twice the standard cellular rates. (2) Current transport solutions treat wireless links essentially as black boxes, making it difficult to promptly respond to the rapid wireless fluctuations [57].

Therefore, effectively addressing wireless disruption demands a multi-path solution that treats wireless as a white box without requiring additional costs for users. Fortunately, commercial 5G already provides such opportunity with *multi-connectivity*[1]–a mechanism that allows a single user equipment (UE) to connect to multiple base stations (BSs) [11] simultaneously. Originally designed to aggregate bandwidth during the early deployment of 5G networks [70], multi-connectivity can deliver multi-path benefits at the link level, enabling faster detection and recovery from link failures compared to higher-layer multi-path transport solutions, while requiring only a single cellular subscription. With proper enhancements focused on reliable transmission, multi-connectivity offers significant potential to improve performance for RTS apps that demand both low latency and high throughput.

However, our measurements on commercial 5G networks reveal that multi-connectivity remains underutilized for RTS apps. As shown in Figure 1, these apps frequently encounter degraded 5G throughput and, consequently, low delivered bitrate, yet the 4G leg is left largely idle in 5G multi-connectivity networks, even when 5G throughput suddenly deteriorates and necessitates compensation from the 4G link. Our further analysis in §3.3 reveals that this issue stems not from insufficient 4G bandwidth, but from operators' deliberate strategy to conserve backup 4G link regardless of the app demands.

[1]Current commercial 5G primarily employs dual-connectivity, a specific form of multi-connectivity that connects 4G and 5G BSs. In this paper, we consider a more general form and refer to it as multi-connectivity.

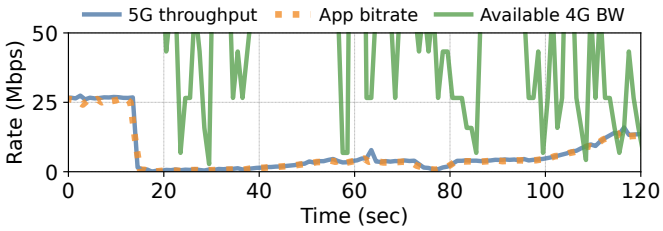


Figure 1: Underutilization of multi-connectivity in commercial 5G networks[2]. When 5G throughput degrades (around 17 s), the RTS bitrate also drops, despite sufficient 4G bandwidth being available through multi-connectivity.

As a result, critical opportunities to improve app QoE through link-level multi-path are wasted.

To address this, we introduce QCON, the first seamless QoE-driven multi-connectivity solution that schedules packets across multiple links based on precise app requirements. With the QoE in mind, QCON can efficiently utilize the backup links only for the app requirements. Importantly, QCON aims to achieve this without requiring app modifications, creating commercial opportunities for 5G operators to support RTS apps while meeting both performance enhancements with multi-link capabilities and users' accessibility needs. However, implementation must overcome the following challenges:

- **How to enable QoE-awareness in the network?** Frame-level transmission information is crucial for understanding the RTS app's QoE, directly affecting frame rate and bitrate [26]. However, such app-level frame information is not explicitly exposed to the intermediate router, such as BS. QCON uses RTS protocol metadata and last-mile BS transmission status to track frame-packet associations, enabling precise frame-level tracking without app modifications.
- **How to schedule packets to maximize video QoE?** Frame-level tracking helps deadline guarantees, but merely ensuring deadline satisfaction, as in prior work [58, 83], can degrade video quality due to RTS congestion control that reduces bitrate according to the delay jitter [48]. QCON employs adaptive deadline-based multi-path scheduling, optimizing link usage and preventing jitter-induced bitrate degradation by fine-tuning deadlines to avoid triggering congestion control at the end host.
- **How to prevent frame stalls during link disruptions?** Preventing frame stalls during link disruptions is essential for RTS QoE. Traditional RAN recovery methods like retransmission [12] and handover [54] are designed for single-connectivity and underutilize multi-connectivity. QCON prevents stalls by intelligently redirecting packets from delayed links to alternative paths based on QoE signals.

[2]To estimate unused 4G capacity without active traffic injection, we calculated available bandwidth using spectral efficiency (derived from SINR (Signal to Interference plus Noise Ratio)) multiplied by maximum resource blocks, measured overnight when background load is minimal.

We prototype QCON on the RAN Intelligent Controller (RIC) within an Open-RAN testbed using OpenAirInterface5G's 5G/4G BSs [4]. Extensive evaluations, including both emulation and real-world experiments on WebRTC and commercial cloud gaming services, show that QCON outperforms existing solutions, boosting tail frame rates by 4-5× with higher bitrates by up to 2.1× with efficient radio resource usages.

Our contributions are summarized as follows:

- We analyze the underutilization of 5G multi-connectivity for RTS app QoE.
- We introduce a novel QoE-driven multi-connectivity framework within an RIC that leverages link-layer multi-path benefits for RTS apps.
- We prototype and evaluate QCON in real-world settings, showing significant QoE improvements for RTS apps in practical deployments.

2 Backgrounds

In this section, we first discuss 5G, which is a key technology to enable high-quality mobile RTS. We then provide a background on WebRTC, the de-facto standards of RTS apps [26, 49].

2.1 5G Primer

5G technology aims to deliver high-bandwidth, low-latency services to meet the demanding requirements of modern mobile apps. Below, we highlight the key 5G concepts essential for low-latency, high-throughput RTS.

Multi-connectivity. Modern 5G networks permit a single UE to connect to multiple BSs simultaneously, a feature known as *multi-connectivity* [66]. A common example is dual-connectivity in 5G non-standalone (NSA) mode, where 5G and 4G links operate in parallel. Crucially, 3GPP standards also extend multi-connectivity to standalone (SA) deployments [11], allowing a UE to concurrently link to multiple 5G cells (e.g., sub-6GHz 5G + mmWave 5G). This capability is distinct from carrier aggregation (CA), which aggregates multiple carriers at a single BS [51], though both technologies can be used in tandem.

Data transmission procedure. Figure 2 shows the data-plane stack for 5G NSA, illustrating the path from the core network down to the physical (PHY) layer. At the top, ❶ the Packet Data Convergence Protocol (PDCP) layer receives IP data packets from the core network and is responsible for link-level multi-path scheduling in multi-connectivity. As shown in the figure, ❷ PDCP performs path scheduling, directing packets to either 5G or 4G radio access networks through Xn interface [11]. Below PDCP are the Radio Link Control (RLC) and Medium Access Control (MAC) layers, which handle packet segmentation, buffering, and radio resource coordination. On the user equipment (UE) side, ❸ incoming packets traverse

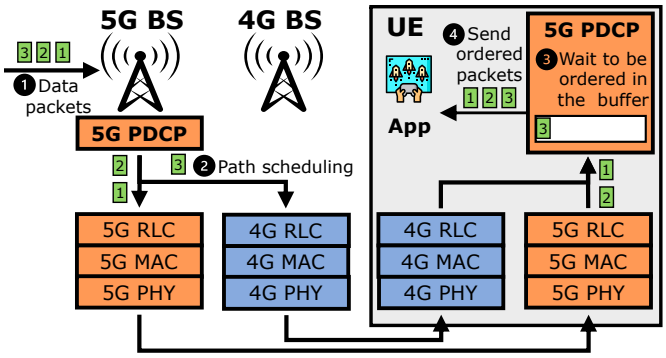


Figure 2: Packet transmission procedure in the data plane of multi-connectivity-enabled 5G. The PDCP layer serves as the anchor for multi-connectivity, handling path scheduling on the sender side and packet reordering on the receiver side.

	OpX	OpY	OpZ
NSA (Non-standalone)	✓	✓	✓
SA (Standalone)	✗	✗	✓
Multi-connectivity enabled	✓	✓	✓

Table 1: Operational support across deployment options. All operators can provide multi-connectivity features.

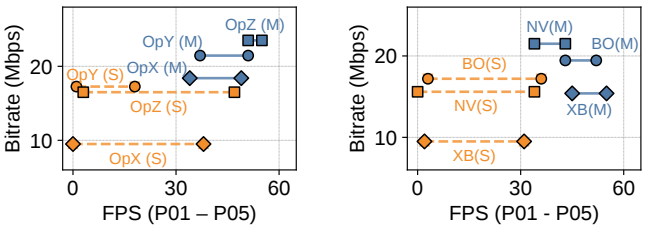
up through the protocol stack where the 5G PDCP layer waits for packets to be ordered in its buffer before delivering them in sequence (1, 2, 3) to the app (❹).

2.2 Overview of WebRTC

WebRTC, as the de facto standard for RTS apps, underpins interactive apps such as cloud gaming [41], video conferencing [1, 6], and teleoperation [25]. In the WebRTC pipeline, senders continuously produce, encode, and transmit video frames, which are decoded by receivers only after all packets for each frame have been received [57, 77].

Requirements. Two critical requirements ensure high Quality of Experience (QoE): 1) Frame-level deadline satisfaction, where each frame must reach receivers within tight deadlines to maintain high frame rates without stalling [57, 58], even at the lower tail, to ensure a consistent user experience [77]; and 2) High throughput, as better video quality requires higher encoding bitrates [32], with cloud gaming, for instance, requiring 25 Mbps for 1080p at 60 FPS [63].

Congestion control algorithm (CCA). To balance high throughput with low latency, WebRTC employs latency-sensitive CCAs that monitor both delay jitter and packet loss, enabling rapid adaptation to network queuing dynamics [20, 57, 84]. These algorithms detect congestion based on jitter patterns across groups of packets (e.g., per-frame), and when jitter exceeds a threshold, congestion is inferred and the sender reduces its bitrate accordingly [48].



(a) Operators. (b) Commercial apps.

Figure 3: RTS apps' performance over commercial 5G that mainly utilize a single link (S), and an commercial traces-based emulation that fully utilizes both 5G and 4G links (M).

3 Motivations

This section presents comprehensive measurement results from current multi-connectivity-enabled 5G networks running RTS apps, revealing systematic underutilization and uncovering key opportunities to enhance RTS performance.

3.1 Multi-Connectivity in Commercial 5G

We investigated multi-connectivity in three commercial 5G networks (OpX, OpY, OpZ)[3] across campus, city, and highway environments. Our workload included 1080p, 60 FPS video streaming (with a maximum bitrate of 25 Mbps) using WebRTC and cloud gaming apps (Nvidia GeForceNow (NV) [63], Xbox (XB) [60], Boosteroid (BO) [18]) on Samsung S22 smartphones receiving video frames from AWS (minimum RTT∼19 ms) or provider servers. Link-level statistics were collected with XCAL diagnostic tools [15]. More details about our experiments are in Appendix §A.

As shown in Table 1, our measurements reveal that all operators support dual connectivity with both 4G and 5G BSs. OpX and OpY consistently operate in 5G NSA mode, while OpZ primarily uses 5G SA mode, with occasional transitions to NSA. When 5G signal quality falls below operational thresholds (e.g., RSRP (Reference Signals Received Power) < -115 dBm), connections temporarily revert to 4G-only, resuming 5G connectivity as conditions improve. We confirm that simultaneous data transfer occurs over both 4G and 5G links in both the downlink and uplink directions.

Observation #1. Commercial 5G operators and COTS UEs support multi-connectivity with both 4G and 5G BSs.

3.2 Poor Performance of RTS App on 5G

Despite the multi-connectivity capabilities of 5G, our measurements reveal that commercial deployments often fail to deliver robust QoE for mobile RTS apps. Figure 3 compares the average bitrate and lower-tail frame rates (P01-P05) of WebRTC running on (i) a commercial 5G network and (ii) an ideal multi-link emulation that aggregates 5G and 4G bandwidth, as measured by Iperf [78]. To ensure reproducibility,

[3] OpX: Verizon, OpY: AT&T, OpZ: T-Mobile.

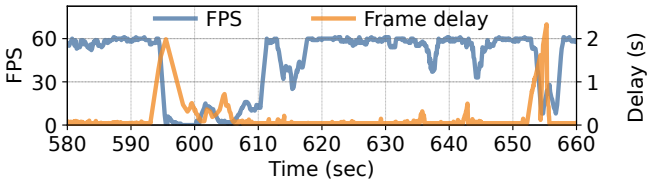


Figure 4: Sudden delay spikes in RTS app significantly degrade frame rates.

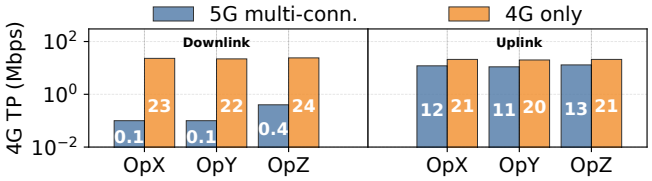


Figure 5: 4G throughput (TP) of RTS apps in 5G multi-connectivity and 4G-only connection. Despite the ability to use 4G and 5G simultaneously in multi-connectivity enabled networks as shown in uplink case, 5G operators mostly prefer to utilize only 5G in the downlink.

the emulation was performed with network traces from 10 iterations of the same driving route with WebRTC experiments. Under commercial service, WebRTC achieves an average of just 16.3 Mbps and drops to 1 FPS at the P01, while the ideal scenario maintains 23.5 Mbps and 51 FPS. Similar patterns are observed in commercial cloud gaming in Figure 3(b), indicating that the issue is agnostic to the specific app.

A more detailed timeline (Figure 4) reveals that transient 5G delay increases—often triggered by mobility-induced handovers or wireless errors [36, 82]—cause sharp frame-delay spikes, notably around 595 s and 655 s. These disruptions are exacerbated by the conservative nature of many RTS congestion control schemes, which typically follow additive-increase, multiplicative-decrease (AIMD) logic [20]. Once a delay spike is detected, sending rates are aggressively reduced and may take several minutes to recover (as in Figure 1), leading to prolonged low-bitrate streaming. These findings highlight the need to mitigate sudden link disruptions with multi-connectivity, to ensure consistent RTS performance over mobile 5G.

Observation #2. RTS apps perform poorly over commercial 5G, even when multi-connectivity features are available to provide reliability through heterogeneous links.

3.3 Underutilization of Multi-Connectivity

Multi-connectivity technology offers a potential solution by leveraging multiple networks when a single link fails. However, commercial deployments rarely utilize 4G connections effectively for downlink traffic.

Operator policies limiting multi-connectivity. In Figure 5, our WebRTC experiments in static scenarios compare 4G-only connections with 5G multi-connectivity (with 4G support

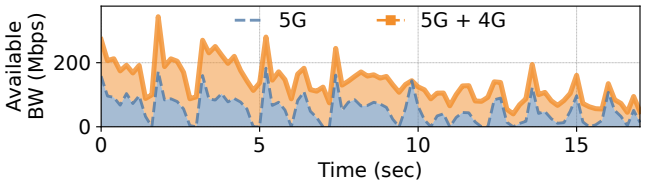


Figure 6: Available bandwidth measured by simultaneous Iperf test on 5G and 4G UEs. 4G link can complement the sudden capacity drop of 5G.

enabled) in both downlink and uplink scenarios. Using the same 4G BS connection, results show the 4G link under multi-connectivity utilizes only 0.1∼0.4 Mbps while standalone capacity is enough to serve more than 20 Mbps in downlink. Meanwhile, the bandwidth-constrained uplink could utilize over 50% of 4G only session throughput. This asymmetry indicates operators deliberately limit 4G downlink usage for 5G-capable devices, likely to preserve 4G resources for users without 5G coverage.

Potential enhancement with multi-connectivity. Despite operator-imposed restrictions, our simultaneous Iperf tests show that 4G links could effectively compensate for 5G disruptions, as shown in Figure 6. In the experiment on OpZ's network, we observe that utilizing both 4G and 5G links yields a P01 throughput of 73 Mbps and a P0.1 throughput of 20.9 Mbps, compared to 0 Mbps at both percentiles when using 5G alone. This is because the throughput of 5G link and the 4G link are weakly correlated (ρ=0.28). We attach the details of our traces that show the potential of multi-connectivity in Appendix A.

Why not simply enable 4G? In principle, operators could seamlessly leverage 4G alongside 5G; however, in practice, they often reserve 4G capacity for devices outside 5G coverage. This system-wide policy avoids overloading 4G links that must serve users lacking reliable 5G signals. Yet this approach is overly cautious from the perspective of RTS apps, which experience abrupt video stalls and latency-sensitive CCA-induced bitrate collapses even with brief 5G lapses. To this end, a more nuanced multi-connectivity scheduler is needed—one that opportunistically invokes 4G resources only when necessary to enhance the app QoE, rather than defaulting to a rigid "4G or 5G" fallback strategy.

Observation #3. Current commercial 5G implementations prioritize preserving 4G capacity and underutilize the multi-connectivity features regardless of the app demands.

3.4 Approach

Existing app-awareness solutions in the RAN either lack visibility into QoE or rely on deployment mechanisms that are impractical. For instance, 5G QoS frameworks [8, 9] manage the RAN based on packet-level service level agreements (SLAs), but cannot capture the frame-level transmission status that is critical to RTS QoE [26]. Recent solutions that do offer frame-

level insights [22, 83, 86] require modifications to both the app and the RAN, necessitating tight coordination between independent entities. To address these limitations, we propose an advanced multi-connectivity scheduler that optimizes RTS QoE without intrusive system changes, achieving high performance and practical deployability.

4 QCON Design

We present QCON, an operator-side, **Q**oE-driven multi-**CON**nectivity framework in the RAN that improves the QoE of RTS apps without requiring any cooperation from the apps themselves. Below, we outline the key design challenges and our considerations for efficient packet scheduling over multi-connectivity.

4.1 Design Challenges

- **Understanding QoE signals while preserving transparency.** RTS QoE is determined by metrics such as video quality and stalling [26], which are not directly observable at the network layer. While app-level solutions [26, 87, 88] implement control channels by modifying both sender and receiver apps, such coordination is infeasible in RAN environments. To remain compatible with unmodified, existing RTS apps, QCON must infer QoE signals in an app-transparent manner.

- **QoE-driven multi-link scheduling.** Even with QoE insights, effective scheduling requires deeper app awareness. Simply optimizing network-level metrics such as packet delay or throughput may not directly improve QoE, and can lead to inefficient use of backup links. A QoE-driven approach must align packet-level decisions with app-level needs.

- **Preventing frame stalls from link disruptions.** Meeting frame deadlines is critical, even during transient link failures. While existing RAN mechanisms such as HARQ retransmission [12] offer some resilience, they rely on a single link and can still fail. BS-to-BS handover solutions [54] provide link-layer continuity but lack awareness of app QoE. QCON must protect frame delivery deadlines by proactively leveraging multi-link redundancy with QoE in mind.

4.2 Frame-Level Insights into App QoE

The main metrics that determine the RTS QoE–video quality and video stalls–are tightly coupled with frame-level transmission dynamics. Bitrate, which determines video quality, is largely influenced by the CCA, which adjusts based on frame-level jitter. Meanwhile, video stalls occur when frames fail to arrive within the app's playback deadline [26]. This implies that while the RAN may not directly access app-level QoE

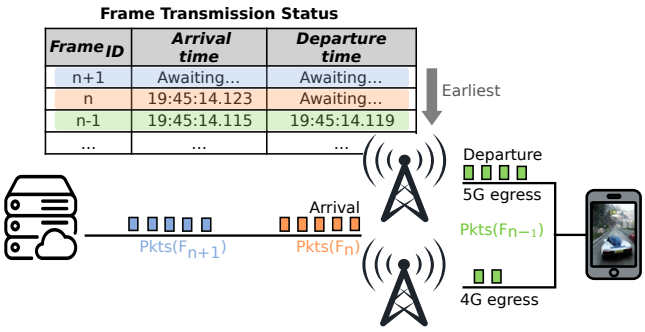


Figure 7: Frame-level transmission status tracking for transparent RTS QoE estimation in RAN, utilizing frame identification from packet metadata and direct link transmission observability.

signals, it can infer them by tracking frame-level transmission behavior. Existing QoE estimation techniques on in-network devices [59, 72] struggle to track accurate frame-level transmission dynamics, as they focus on high-level and straightforward metrics, such as frame rates, that fail to capture varying frame transmission patterns induced by pacing [38] or encoding [44]. Notably, we observe that RTS packet metadata contains rich implicit information about their frame membership, which can be exploited for accurate frame-level tracking (§5.2). As the last-hop router, the RAN's BS holds a uniquely privileged vantage point to observe packet arrivals at the UE. As shown in Figure 7, combining this vantage with per-packet frame membership insights can reconstruct frame transmission timelines with millisecond-level accuracy.

4.3 Frame-Aware Multi-Link Scheduling

Given access to frame-level transmission status, multi-link scheduling should intelligently leverage multiple links to optimize app performance. However, to be effective, the scheduling design must account for the specific characteristics of real-time video. We introduce two key design strategies in our multi-link scheduler:

Frame-level deadline-aware scheduling. Naively distributing packets across multiple links to optimize low-level metrics (e.g., delay or throughput) can lead to unnecessary use of backup links without tangible improvements in app performance. To maximize efficiency, our scheduler activates the backup link only when it anticipates an improvement in app QoE. This is achieved by enforcing SLAs on a frame-by-frame basis, ensuring that each video frame is delivered within its deadline using the minimum necessary resources.

Jitter-adaptive deadline adjustment. Meeting deadlines alone is insufficient. Even when frames arrive within their nominal deadlines, they can trigger unintended CCA responses if delivery timing introduces high jitter. For example, a pair of frames arriving at 10 ms and 90 ms may technically meet a 100 ms SLA [14], but such uneven arrival times can mislead latency-sensitive CCAs into interpreting congestion,

prompting them to reduce bitrate unnecessarily. Our key insight is that frame-level jitter significantly influences RTS CCA behavior [20, 48], and since we track frame-level transmission, we can observe this jitter directly. Leveraging this, our scheduler dynamically adjusts frame deadlines to suppress jitter spikes and avoid CCA-triggered bitrate degradation.

4.4 Priority-Based Multi-Link Re-Injection

Even when frames are carefully scheduled over multi-connectivity using current channel status, unpredictable wireless variations can still lead to deadline violations. Rather than attempting to predict future channel conditions, which is inherently uncertain, we instead exploit the heterogeneity of multiple links to proactively protect frames from stalling. A common approach in transport-layer multi-path systems is packet re-injection, where packets experiencing delay are redundantly sent over alternate paths to meet deadlines [47, 87, 88]. However, these solutions treat underlying links as opaque and often rely on brute-force duplication, which wastes radio resources and lacks urgency awareness for high-priority video frames. In contrast, QCON introduces a more efficient and QoE-aware re-injection mechanism by leveraging existing RAN capabilities. Specifically, we repurpose the packet-forwarding technique traditionally used for BS handovers [54], which moves packets from a source cell to a target cell to avoid packet loss. QCON uses RAN-level controllability to selectively re-inject urgent frames based on their deadlines, ensuring that they bypass queuing delays and meet tight app timing requirements.

5 QCON System

This section provides an overview of QCON, including its system architecture and core modules.

5.1 System Overview

Figure 8 illustrates QCON implemented within the real-time RIC of Open-RAN [28, 45]. This architecture comprises three key modules that interface with Open-RAN's functional split components: the Central Unit (CU), which hosts the Packet Data Convergence Protocol (PDCP) layer, and the Distributed Unit (DU), which manages the lower protocol stack including RLC, MAC, and PHY [64]. These modules exploit multi-connectivity capabilities at the link level to enhance app QoE in app-transparent way. Note that QCON supports only registered apps, identified via 5QI (5G QoS Indicator) [14] that can differentiate traffic flows.

QoE Monitor profiles frame transmission by tracking packet arrival via metadata in RTS packets (i.e., Real-Time Transport Protocol (RTP) [79] headers) and departure status through link transmission status. This allows it to assess real-time frame delivery progress and inter-frame jitter.

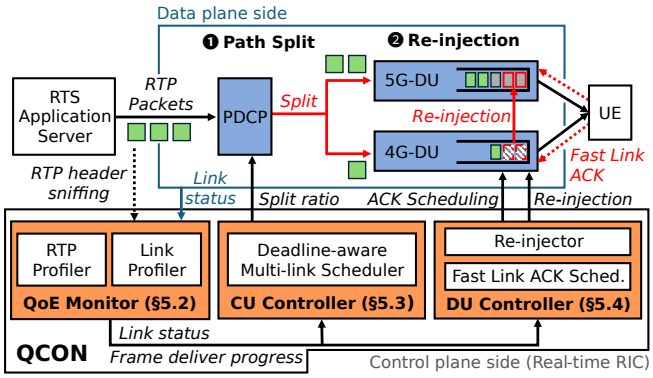


Figure 8: System overview of QCON.

CU Controller manages the CU's PDCP layer to optimize packet distribution across multiple links based on frame information and link status given by QoE Monitor. It selects paths that meet frame-level deadlines while minimizing reliance on backup links. To maintain consistent bitrates, it dynamically adjusts deadlines in response to latency fluctuations.

DU Controller provides two main functionalities: (a) Fast link ACK technique for immediate feedback on frame delivery, and (b) Rapid packet re-injection, which reroutes delayed packets to faster available links using packet forwarding in handover mechanisms, with priority scheduling for time-sensitive frames.

5.2 QoE Monitor

QoE Monitor captures QoE signals of RTS apps by detecting frame delivery status in the RAN. It profiles RTP headers for arrivals and monitors link status for departures, providing comprehensive visibility into frame transmission. Here we mainly focus on video frames, but audio frames are equally applicable as they also have RTP headers.

RTP header profiling. To determine which video frame each packet belongs to, QoE Monitor analyzes the RTP headers of incoming packets. When an IP packet arrives at the CU from the internet, the RIC first inspects it to check if it belongs to a registered RTS flow. For identified RTS packets, QoE Monitor extracts two key fields from the RTP header: (1) *RTP timestamp*, which is used for playback synchronization at the receiver. All packets belonging to the same frame share the same timestamp, enabling QoE Monitor to group them accordingly; and (2) *Marker bit*, which indicates the end of a video frame, providing clear frame boundaries. This header information is originally designed to allow receivers to make frame-level decoding decisions without decrypting the entire payload, and is thus exposed even in widely-used secure RTP (SRTP) [80].

The extracted information used in RIC is shown in Figure 9. This includes the packet's timestamp identifier, total frame size in bytes, number of bytes acknowledged via link-layer ACKs or dropped bytes (e.g., from buffer overflow), marker

```
struct rtp_frame_info{
    int timestamp;
    int frame_size;
    int acked_bytes;
    bool
    ↪ marker_received;
    int64_t
    ↪ arrival_time;
    int64_t
    ↪ departure_time;
    int64_t deadline;
};
```

Figure 9: Frame information.

```
struct DU_stats{
    int path_id;
    int mac_total_RBs;
    int mac_TBS;
    int mac_RBs;
    int rlc_acked_bytes;
    int rlc_dropeed_bytes;
};
struct CU_stats{
    int path_id;
    int sent_bytes;
};
```

Figure 10: RAN information.

reception status, arrival time of the first packet in the frame, and the frame's departure time from the network. Additionally, we set the frame's deadline based on the given SLA (e.g., 100 ms), along with the arrival time of the first packet at the RAN. During transmission, QoE Monitor continuously reports frame progress to DU Controller to evaluate potential re-injection triggers. Upon complete acknowledgment of all packets within a frame, this metadata is forwarded to CU Controller to inform subsequent scheduling decisions.

Link status profiling. After extracting frame-level information from RTP headers, QoE Monitor collects RAN transmission statistics to determine the current link status and track frame departures, as summarized in Figure 10. It monitors link capacity and buffer occupancy using Open RAN's Key Performance Metrics (KPMs) [30], enabling accurate frame delivery time estimation in a standards-compliant manner.

To track packet transmission status, QoE Monitor leverages RLC-layer KPMs, as the RLC layer—particularly in Acknowledged Mode (AM)—offers precise in-order delivery statistics via link-level ACK mechanisms [74]. It monitors buffer status upon RTS packet enqueueing (sampled at 0.5 ms intervals) and tracks transmitted bytes upon ACK reception. However, commercial 5G ACK reporting mechanisms introduce substantial delays (>100 ms), which hinder real-time tracking of frame progress. To mitigate this, we introduce a fast link ACK technique, detailed in §5.4.

Additionally, QoE Monitor performs bandwidth monitoring to enable accurate multi-link scheduling in the future. For this, QoE Monitor subscribes to MAC-layer KPMs and calculates the available bandwidth on each link using:

$$\mathrm{BW}_{i,k} = \mathrm{SE}_{i,k} \cdot \left(R_{i,k} + \frac{R_{\max,k} - \sum_{j \neq i} R_{j,k}}{N_k}\right), \tag{1}$$

where $\mathrm{SE}_{i,k}$ is the measured spectral efficiency for user i on link k, $R_{i,k}$ is the number of resource blocks allocated to the user, and N_k is the total number of active users on link k [82]. This direct bandwidth estimation from the RAN offers more accurate and responsive insights compared to indirect transport-layer methods. QoE Monitor monitors these metrics continuously, performing bandwidth calculations each time the first packet of a new frame is detected.

5.3 CU Controller

CU Controller determines packet distribution across multiple wireless links to maximize RTS app performance. It operates through two coordinated mechanisms.

Deadline-aware multi-link scheduling. CU Controller selects the optimal set of links and the corresponding split ratio to meet delivery deadlines with minimal cost, based on the most recent link status. Unlike in multi-path transport, where the data and control paths are managed on a per-packet basis, allowing packets to arrive at the scheduler and depart immediately after scheduling, RICs operate in a near real-time regime for control and data paths. As a result, the multi-link scheduler must make decisions before frames arrive. To address this, CU Controller uses frame arrival information to ensure scheduling is performed with the most up-to-date view of both link and frame conditions. Specifically, the scheduler observes frame arrival patterns (i.e., frame rates) over a 100 ms window and proactively schedules transmission half a frame cycle ahead of expected arrivals, using the average recent frame size as an estimate. Within this setting, the scheduler solves the following optimization problem (We omit UE index i for simplicity):

$$\min_{S \subseteq K} \sum_{k \in S} c_k,$$

$$\text{s.t.} \quad \max_{k \in S}\left(d_k + \frac{s_k}{\mathrm{BW}_k}\right) \leq D_m, \quad \sum_{k \in S} s_k = f_m, \quad d_k = \frac{B_k}{\mathrm{BW}_k} + l_k. \tag{2}$$

Here, f_m is the estimated size of the m-th frame, BW_k is the available bandwidth on link k, and s_k is the portion of the frame routed over link k. The term d_k captures queuing and fixed latency, with B_k the current buffer occupancy and l_k the per-link interface delay. Each link incurs a cost c_k (e.g., 1 for the primary 5 G link and 2 for the backup 4 G link). The scheduler enumerates link subsets $S \subseteq K$, selecting the lowest cost set that satisfies the deadline D_m; if none do, it chooses the subset with the smallest expected completion time. For multi-link solutions, the split ratio $\alpha_k = s_k / f_m$ is obtained by equalizing finish time across all selected links [62].

Jitter-adaptive deadline setting. We dynamically adjust frame deadlines instead of applying fixed SLA latency limits. This prevents RTS apps from reducing bitrate when they misinterpret jitter as congestion. However, the challenge is determining the appropriate jitter threshold without unnecessarily using backup links, while modern RTS apps employ proprietary CCA implementations [48, 56].

Our solution derives this threshold from two sources: 1) *Offline measurements:* For different RTS apps, we measure the maximum tolerable frame-level jitter J_{off} using methodology from [48]. By running the apps on an emulated network and observing their reactions while incrementally increasing jitter, we determine app-specific jitter thresholds. 2) *Online adaptation:* CCA sensitivity varies with network conditions [20], so we continuously monitor frame jitter and bitrate. If an app

Algorithm 1: Deadline-aware multi-link scheduling

Input: Estimated $m-th$ frame size f_m; link set K SLA latency $D_{\max}$; jitter threshold J_{th}
Output: Chosen link set $\mathcal{S}^\star$ and split ratios $\{\alpha_k\}$

1 $D_m \leftarrow \text{AdaptiveDeadline}(J_{\text{th}}, D_{\max})$;
2 $\mathcal{S}^\star \leftarrow \varnothing$;
3 **foreach** $\mathcal{S} \subseteq K$ *in ascending cost* **do**
4 $T \leftarrow \text{EqualFinishTime}(\mathcal{S}, f_m)$;
5 **if** $T_{\max} \le D_m$ **then**
6 $\mathcal{S}^\star \leftarrow \mathcal{S}$;
7 **break**;
8 **foreach** $k \in \mathcal{S}^\star$ **do**
9 $s_k \leftarrow \text{ChunkSize}(k, T)$;
10 $\alpha_k \leftarrow s_k / f_m$;
11 $\text{SendCuCommand}(\{(k, \alpha_k) \mid k \in \mathcal{S}^\star\})$;

demonstrates reducing bitrate at smaller jitters J_{on}, we update the threshold to J_{on} from J_{off}. Specifically, we update J_{on} to J_{th} if bitrate degrades by 15% or more (based on [48]) due to jitter lower than J_{off} within the past 500 ms. If no subsequent bitrate degradation occurs, we gradually increase J_{on} by 1 ms every 500 ms for efficient backup link utilization.

Our scheduler uses this adaptive threshold to set frame deadlines. For the m-th frame (with RAN arrival time t_m and expected departure time T_m), the deadline is:

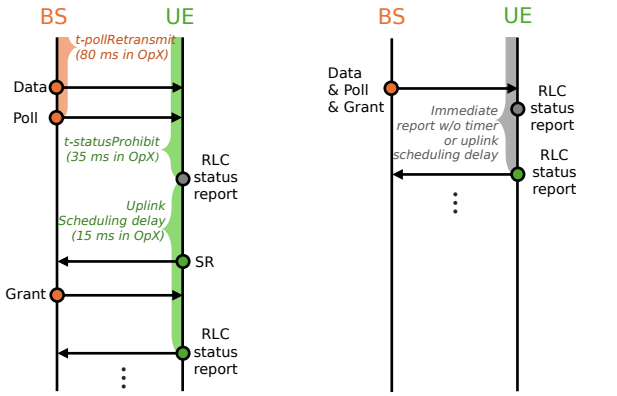
$$D_m = \min\Big\{D_{\max},\ \big(T_{m-1} + (t_m - t_{m-1}) + J_{\text{th}}\big)\Big\}, \quad (3)$$

where D_{max} represents the app's SLA latency requirement, ensuring that meeting the deadline satisfies the SLA. The workflow of our multi-link scheduling is described in Algorithm 1. Upon each scheduling trigger that occurs just before frame arrival, QoE Monitor computes the deadline based on recent frame transmission status and the jitter threshold (line 1). Using this deadline, the scheduler determines the optimal combination of links capable of delivering the frame within the deadline while minimizing link utilization costs (lines 2-10). Then, QoE Monitor sends a control command to the CU with the link set and frame split ratio (line 11).

5.4 DU Controller

DU Controller operates at the DU level to ensure timely reporting of frame delivery status and rapid recovery from link disruptions. It implements two primary mechanisms:

Fast link ACK mechanism. Previous researches to determine the transmission status of app data units [83, 86] have simply utilized the estimated bandwidth at the MAC layer, but this cannot track out-of-order packets that can occur due to channel errors [74], making it difficult to know exactly when each packet is sent in the frame. To accurately track the real-time frame delivery progress, we leverage the existing 5G ACK reporting mechanism in RLC layer, which is originally designed to notify the out-of-order packets for retransmission.



(a) Commercial 5G mechanism. (b) Fast link ACK mechanism

Figure 11: Compared to commercial 5G with high link ACK delay, our mechanism reduces ACK report delay through optimized timer settings with immediate poll and uplink grant.

Unfortunately, current 5G ACK reporting mechanism is insufficient for real-time tracking of frame delivery progress due to its reliance on conservative RLC ACK reporting. This delay hinders the responsiveness of the scheduling algorithm. As shown in Figure 11(a), the delay arises from the inherent RLC timers at both the BS and UE [74], and from uplink scheduling protocols that require grant allocation before transmission [75, 83]. Specifically, the BS's *t-pollRetransmit* timer in OpX activates every 80ms to check for retransmissions, while the UE's *t-statusProhibit* timer (35ms in OpX) limits the frequency of status reports. Furthermore, uplink packets can take up to 15 ms to be scheduled due to the SR (Scheduling Request) process, which occurs periodically (every 10 ms in OpX) when the UE buffer is empty. In the worst-case scenario, these delays accumulate, and it can take up to 130 ms for the BS to receive an ACK from the UE.

To address these delays, DU Controller implements an optimized link ACK mechanism. As illustrated in Figure 11(b), this technique disables conventional RLC timers for RTS UE and instead immediately triggers RLC ACKs using poll bits during data transmission. Simultaneously, uplink grants are proactively provided to minimize scheduling latency for returning status reports. This method incurs slightly more radio resource usage than general-purpose 5G, but the overhead is minimal: RLC status reports are under 8 bytes [13], and uplink grants to transmit these consume less than 100 bytes—negligible in the context of high-bitrate RTS.

Priority-aware re-injection. When network conditions deteriorate unexpectedly, DU Controller rapidly migrates packets to alternative links to prevent deadline violations. Unlike traditional transport layer re-injection methods, DU Controller leverages RAN's packet forwarding capabilities, typically used during handovers, to transfer buffered packets between DUs without duplication, significantly improving radio resource efficiency.

The re-injection process is triggered when DU Controller

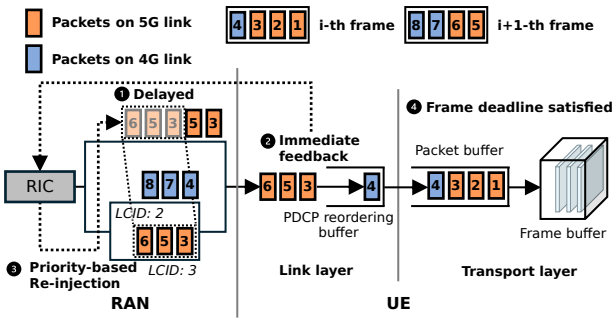


Figure 12: Our re-injection immediately detects the delay and forwards frames to the alternative link, considering its priority.

detects that the expected delivery time for a frame's remaining packets calculated by recent bandwidth status exceeds the available slack time (deadline minus elapsed time) by a factor of (1+β), where β is a configurable tolerance parameter. A higher β results in more frequent re-injections to prevent video stalls but may increase backup link utilization. Once triggered, DU Controller solves a scheduling problem to identify the optimal alternative links and forwards unacknowledged packets from the disrupted link accordingly. If there is no available set that satisfies the deadline, we do not re-inject the packet since this would rather increase the delay of alternative links.

If the stalled frame has a shorter estimated delivery time than the fastest frame on the alternative link, it is placed in a priority queue to ensure faster delivery. To achieve this, DU Controller leverages 5G's logical channel ID (LCID) priority mechanism [12]. By assigning higher priority LCIDs to re-injected packets, they can bypass lower-priority traffic in RAN queues. This ensures faster delivery than methods relying on transport layer mechanisms or standard packet forwarding that do not account for app-level QoE requirements. As shown in Figure 12, this mechanism enables quick service of delayed packets: ❶ when video stalls occur, ❷ RIC receives immediate feedback on frame delivery progress, ❸ issues a re-injection command to the underperforming link, and prioritizes these packets on the alternative link to be delivered on deadline (❹).

6 Implementation

We implement our system based on OpenAirInterface5G [4]. While OpenAirInterface5G supports 5G NSA mode, the existing implementation only enables data transmission via 5G link. We extend this implementation to support simultaneous data transmission over both 5G and 4G links. Our modifications to OpenAirInterface5G comprise approximately 2 k lines of new/modified C++ code implementing dual-connectivity features and RIC interfaces. These include data split functions at the PDCP layer and Xn interfaces between 5G and 4G BSs for coordinated data transmission.

QCON is developed as a μApp on EdgeRIC [45], an Open-RAN-compliant RIC framework enabling millisecond-scale control by co-locating with RAN device. Following EdgeRIC's implementation, the interfaces between RIC and RAN are implemented with ZMQ interface [81]. This includes 10 k lines of new C++ code.[4] Specifically, QoE Monitor's packet profiling leverages eBPF [29] to filter and capture packets with minimal overhead. This allows direct sniffing of incoming RTP packets with TC INGRESS FILTER [53] from the network interface without RAN modification. Additionally, to implement DU Controller's re-injection technique, which adopts the data forwarding mechanism in handover, we utilize the data interface implemented on 5G and 4G BSs for dual-connectivity feature.

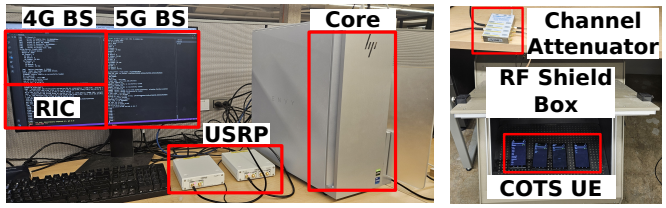


Figure 13: Multi-connectivity enabled Open-RAN testbed.

7 Performance Evaluation

7.1 Experimental Testbed

Open-RAN testbed. For evaluation, we construct an end-to-end Open-RAN testbed with OpenAirInterface5G 5G/4G BSs and Magma core network [3] as shown in Figure 13. Hardware includes two USRP B210 radio units [69] and a Linux desktop (CPU of i7-12700, 32 GB RAM). We use four COTS Android 14 smartphones (three Pixel 6a, one Pixel 7 Pro) with programmable sysmoISIM-SJA2 SIM cards [73]. The network operates in NSA mode: 5G with 20MHz bandwidth (n78 band), numerology 1, single MIMO layer, and 5DDDSU TDD duplex; 4G with 20MHz bandwidth (b41 band) in FDD mode. Our setup achieves up to 73 Mbps on 5G and 37 Mbps on 4G. Experiments run in an RF shield box with Mini-Circuits programmable channel attenuator [5] to emulate real-world highway driving SINR traces over OpX. We setup Xn interface delay between CU and each DU as 10 ms considering Open-RAN's disaggregated architecture [10].
Trace-driven emulator. Since the testbed hardware setup provides limited bandwidth compared to the commercial 5G, we supplement our evaluation using an emulator with real-world 5G traces (Table 5 in Appendix). Implemented with 11 k lines of C++ codes, this emulator leverages Linux's TUN interface [27] to process internet traffic and forward it through emulated cellular protocol stacks (PDCP/RLC/MAC/PHY).

7.2 Evaluation Setup

Apps. Experiments include various RTS apps: open-source WebRTC [33] and commercial cloud gaming (Nvidia

[4]We release the source code of QCON at [46].

Alg.	OpX					OpY				
	VMAF	Bitrate	P05/P01 FPS	4G Usage	4G TP/Total TP	VMAF	Bitrate	P05/P01 FPS	4G Usage	4G TP/Total TP
Single	58.3±11.6	3±2.1Mbps	42/0	0Mbps	0%	92.3±12.3	20.5±7.6Mbps	37/8	0Mbps	0%
minRTT	54.5±8.8	2.8±1.9Mbps	48/11	1.7Mbps	60%	83.7±16.4	13.5±5.3Mbps	32/15	5.6Mbps	41.4%
DCH	59.2±12.3	3.1±2.1Mbps	41/15	1.5Mbps	47%	85.2±14.7	15.3±4.8Mbps	34/12	4.5Mbps	29.4%
PD	90.2±7.3	19.1±5.3Mbps	50/36	19.1Mbps	100%	93.5±9.5	22.1±3.2Mbps	49/44	22.1Mbps	100%
5QI	74.3±14.4	9.5±6.3Mbps	38/8	3.2Mbps	33%	87.1±15.7	17.3±4.0Mbps	40/18	2.3Mbps	13%
QCON	**91.5±7.7**	**20.3±4.5Mbps**	**53/38**	4.9Mbps	24%	**94.8±6.3**	**23.3±3.6Mbps**	**53/49**	2.7Mbps	11%

Table 2: Performance comparison of different algorithms in the Campus traces for OpX and OpY. TP is a throughput.

GeForceNow [63], Xbox cloud gaming [60], an Boosteroid [18]). All configurations use 1080p/60 FPS video with a maximum 25 Mbps bitrate, except Xbox, which uses 1080p/60 FPS streaming with a maximum 20 Mbps bitrate.

Baselines. We evaluate QCON with following baselines.

- **Single-connectivity (SINGLE)**: Commercial 5G approach using only primary 5G link.
- **MINRTT** [67]: Multi-path scheduler sending packets on the lowest RTT path, implemented using RLC layer ACK.
- **Packet duplication (PD)** [68]: Reliability-focused technique by duplicating packets across multiple path.
- **DChannel (DCH)** [71]: Recent L3 multi-path scheduler combining high-bandwidth eMBB and low-latency URLLC channels for short flow (e.g., web loading), implemented with 5G as eMBB and 4G as URLLC.
- **5QI** [14]: Modified 5G QoS indicator guaranteeing 100 ms packet-level latency for gaming apps, scheduling packets to backup links only when SLO requirements are at risk.

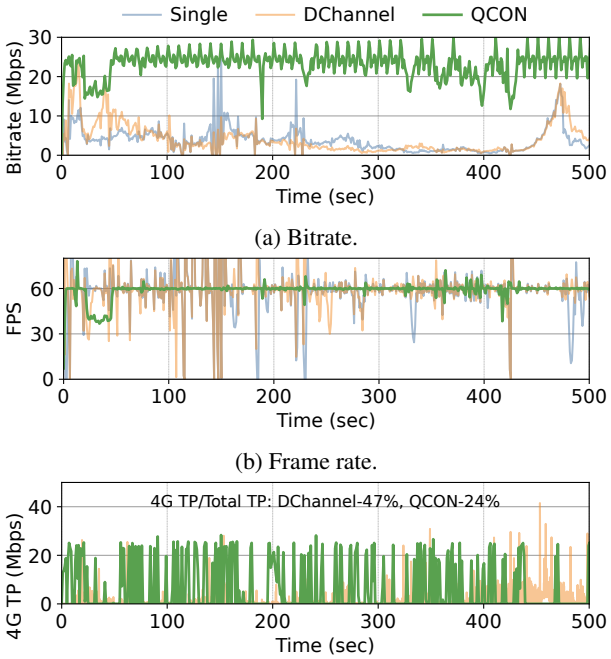
Metrics. Evaluation uses the following performance metrics. 1) *Bitrate:* Average received video rate (Mbps). 2) *Frame rate:* Frames rendered per second, affecting motion smoothness. 3) *VMAF [52]:* Netflix's perceptual video quality metric (0-100). 4) *Normalized performance*. Bitrates, FPS, and backup link usages are normalized with the maximum bitrates, maximum FPS, and the total backup link capacity, respectively.

Evaluation. We evaluate QCON with the following aspects:

- **App performance improvements (§7.3).** We present QCON can improve bitrates up to 2.1× and P01 frame rate by 4-5× compared to the state-of-the-art multi-link scheduling, while preserving less 4G link utilization.
- **Compatibility with various RTS apps (§7.4).** QCON achieves better app QoE in commercial cloud gaming apps without modifying these apps.
- **Microbenchmarks (§7.5).** Microbenchmarks of individual QCON components reveal the specific contributions of each module to overall performance gains.
- **Overheads (§7.6).** QCON shows negligible overheads in terms of CPU, memory usages, and RIC communications.

7.3 Trace-Driven Emulation

We first show the overall performance of QCON in Table 2 over traces on Campus across OpX and OpY. On both traces, QCON achieves the higher bitrates and tail FPS compared to the baselines with comparable 4G link usage. Specifically,



(a) Bitrate.

(b) Frame rate.

(c) 4G capacity usage. TP is a throughput.

Figure 14: Performance of QCON over OpX trace.

QCON shows 2.1× higher bitrates compared to 5QI while enhancing tail P01 FPS from 8 to 38 over OpX trace. While the baselines vary in performance depending on the nature of the trace, QCON is always able to maintain good performance. For instance, SINGLE shows high bitrates in OpY (20.5 Mbps) trace, but achieves only 3.0 Mbps bitrate in OpX trace. This is because OpX trace's 5G link is highly variable, and it is not enough to serve RTS app with only a single link. Even though the average bitrate is comparably high, the tail FPS of a single link is consistently lower compared to the other techniques.

At the same time, existing multi-path solutions show worse bitrate performance than SINGLE, suggesting that simply reducing packet-level delay is not enough to optimize RTS QoE. Similarly, 5QI, which utilizes multi-connectivity without considering frame-level insight, fails to achieve high performance. These techniques mostly utilize lower absolute 4G capacity compared to QCON, but 4G throughput as a percentage of total used throughput is higher than QCON. This shows that QCON efficiently utilizes valuable 4G resources for QoE of RTS app. PD performs the second best after QCON, but this

Alg.	Nvidia GeForceNow [63]			Xbox cloud Gaming [60]			Boosteroid [18]		
	Bitrate	P05/P01 FPS	4G Usage	Bitrate	P05/P01 FPS	4G Usage	Bitrate	P05/P01 FPS	4G Usage
Single	13.2±3.5 Mbps	40/0	0 Mbps	9.5±3.8 Mbps	36/1	0 Mbps	16.5±5.9 Mbps	33/3	0 Mbps
minRTT	9.7±4.0 Mbps	35/10	4.2 Mbps	7.2±3.2 Mbps	21/7	2.5 Mbps	8.1±4.6 Mbps	12/3	4.3 Mbps
DCH	11.1±3.3 Mbps	42/14	2.6 Mbps	7.7±2.9 Mbps	33/3	1.6 Mbps	11.3±5.0 Mbps	37/5	3.6 Mbps
PD	19.4±5.1 Mbps	49/34	19.4 Mbps	16.2±3.1 Mbps	48/45	16.2 Mbps	19.1±3.4 Mbps	49/42	19.1 Mbps
5QI	10.8±5.7 Mbps	37/9	3.5 Mbps	10.4±3.9 Mbps	39/17	7.5 Mbps	12.6±5.5 Mbps	41/19	4.9 Mbps
QCON	**19.8±4.2 Mbps**	**52/35**	5.7 Mbps	**17.9±3.3 Mbps**	**52/48**	7.0 Mbps	**22.8±4.7 Mbps**	**56/50**	5.5 Mbps

Table 3: Performance of three cloud-gaming apps on the OpX's highway SINR trace (standard deviations shown).

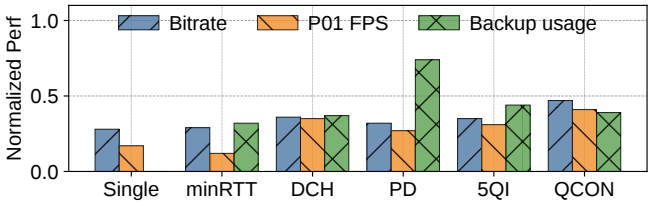


Figure 15: Performance of QCON with multiple UEs.

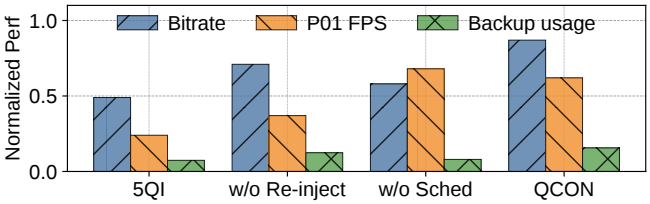


Figure 16: Ablation studies with QCON modules.

is not good in terms of radio efficiency as it requires the same amount of backup link usage as bitrate, and always relies on the performance of the best single link while missing the opportunity of bandwidth aggregation of multi-path.

Figure 14 depicts how QCON gets such consistently high performance. When SINGLE and DCHANNEL experience bitrate degradation or frame rate degradation, QCON actively utilizes 4G throughput. However, DCHANNEL, without app QoE awareness, uses the backup link even during periods without performance degradation (430 s∼470 s). While QCON consumes a higher absolute 4G throughput (4.95 Mbps vs. DCHANNEL's 1.48 Mbps), it uses proportionally less 4G compared to its total throughput—only 24% of total throughput compared to DCHANNEL's 47%. This highlights QCON's efficient backup link utilization for QoE improvements.

7.4 Real-World Experiments

We next evaluate QCON in real-world scenarios.

Performance of commercial apps. QCON's benefits extend to widely used commercial apps. Table 3 shows QCON's performance across three commercial cloud gaming platforms of single UE. QCON consistently achieves high performance regardless of app type in our real-world testbed. These apps are fully compatible with QCON as they utilize RTP protocol and WebRTC-like CCA [56]. Since different apps exhibit varied CCA behaviors under identical channel conditions, QCON's adaptive jitter-aware deadline mechanism allocates more backup link resources to apps more sensitive to jitter. For example, Xbox cloud gaming showed the lowest bitrate performance with SINGLE (9.5 Mbps compared to 16.5 Mbps of Boosteroid), prompting QCON to allocate more backup link resources to maintain high QoE.

Performance under contending UEs. To evaluate QCON in multi-UE environments, we simultaneously ran 4 WebRTC UEs and calculated normalized metrics for average bitrates, tail FPS (across all frame rate samples), and total backup link usage. Figure 15 shows that QCON efficiently utilizes backup links while maximizing QoE. Unlike our single-UE experiments, we found that PD achieves significantly lower normalized bitrates (0.32 vs. QCON's 0.47). As contention on the primary link increases, efficient use of the backup link becomes critical to prevent video stalls during transient congestion; however, PD overutilizes the backup link unnecessarily. We further analyze the impact of QCON on fairness with other UE in Appendix B.

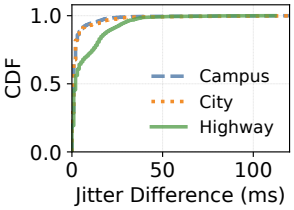


Figure 17: Gap between estimated jitter from QoE Monitor and true value.

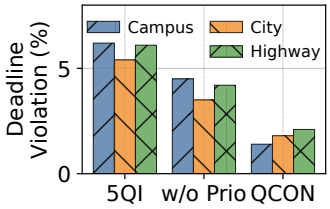
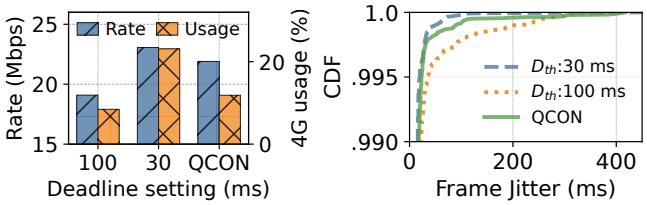


Figure 18: Impact of re-injection technique in deadline violation.

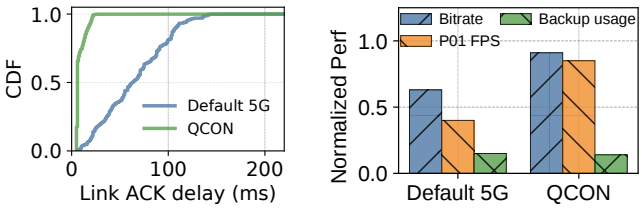
7.5 Microbenchmarks

Ablation studies. To verify each module's effectiveness in QCON, we conduct ablation studies in Figure 16 with two additional comparisons: 1) *w/o Re-inject*, which excludes re-injection module and 2) *w/o Sched*, which excludes deadline-aware multi-link scheduler module. Operating as 5QI without any QCON modules reduces average bitrates by 45% with degraded tail FPS. Adding just the scheduler (*w/o Re-inject*) increases bitrates by 1.4× and enhances tail FPS compared to 5QI. While the re-injection-only configuration (*w/o Sched*) doesn't improve bitrate beyond baseline, it significantly boosts normalized tail FPS from 0.37 to 0.68. The complete QCON implementation shows slightly lower tail FPS than *w/o Sched*



(a) Bitrate and 4G usages. (b) Frame-level jitter.

Figure 19: The impact of multi-link scheduling.



(a) ACK delay improvement. (b) Performance improvement.

Figure 20: The impact of fast link ACK technique.

because increased bitrate introduces additional delay. Nevertheless, full QCON achieves 1.5× higher bitrate than *w/o Sched* and 1.8× higher than 5QI through its effective jitter-aware multi-path scheduling mechanism.

Accuracy of frame-level transmission tracking. Figure 17 shows the CDF of error between QoE Monitor's estimated frame-level jitter and ground-truth jitter from the UE. Across OpX's traces, QoE Monitor accurately tracks frame-level jitter. In campus and city, 94.4% an 93% of estimations show errors below 10 ms, respectively. Even in more variable highway trace, 73% of estimations remain under 10 ms error, and 99% of samples stay below 40 ms.

Priority-based re-injection. Figure 18 measures the deadline (100 ms in our settings) violation ratios for overall frame transmission. We compare our technique with 5QI and W/O PRIO, which re-injects frame without the priority given by LCID. Across three scenarios, our prioritized re-injection achieves violation rates only 1.4%, 1.8%, and 2.1%, outperforming both comparisons which experience more than 2× and 3× higher violation rates for 5QI and W/O PRIO, respectively.

Adaptive deadline-aware multi-link scheduling. To quantify gains from jitter-adaptive deadline-aware scheduling, we compare against two fixed deadlines ($D_{th} = 30, 100$ ms) in OpX's campus trace. As shown in Figure 19(a), shorter deadlines (30 ms) achieve high bitrate but require substantial backup link usage (23% from overall 4G capacity). Looser deadlines (100 ms) reduce backup usage to 8% but decrease bitrates by 18%. QCON's adaptive approach balances these extremes, reducing 4G link usage by 49% compared to the 30 ms setting with only 5% reduced bitrate. This efficiency stems from reduced frame-level jitter at the tail. Figure 19(b) demonstrates that QCON achieves low tail jitter comparable to the 30 ms fixed deadline, while the 100 ms fixed deadline exhibits higher tail frame jitter, which results in lower bitrate.

# of UE	100	200	300	400	500
CPU util (%)	7.2	7.9	8.4	8.6	9.0
Memory usage (MB)	149.6	155.3	161.1	163.7	166.0

Table 4: System-level overheads.

Fast link ACK technique. Figure 20 illustrates the impact of fast link ACK technique. In Figure 20(a), we find fast link ACK technique significantly reduces the link ACK delay. Compared to DEFAULT 5G, which averages 62 ms delay, QCON achieves 6 ms of average delay. This creates a faster control path for our core modules, resulting in performance improvements shown in Figure 20(b): 1.4× higher bitrate and 2.1× better tail FPS. Without fast status reporting, both frame-level jitter and delivery progress information become outdated, compromising the effectiveness of our key modules.

7.6 System-Level Overheads

Finally, we report the system-level overheads of QCON.

CPU and memory overheads. To evaluate QCON's resource requirements, we generated dummy RTP flows (each 10 Mbps) on a local host that QCON processes as UE traffic. Using the LINUX PS tool [7], we found that QCON maintains minimal system-level overheads. Even when handling 500 UEs simultaneously, CPU utilization remains at just 9.0% with memory consumption of 166 MB. This efficiency stems from QCON's use of eBPF for packet inspection and its lightweight control message exchange with RIC.

RIC communication overhead. Communication between RIC and CU/DU components introduces negligible overhead. RAN KPM signaling (Figure 10) requires only 512Kbps of bandwidth when this information is exchanged every slot (i.e., 0.5 ms in 5G). Since our implementation co-locates RIC with RAN components on the same device, this overhead has no measurable impact on QCON's performance.

Power consumption of QCON is reported in Appendix C

8 Discussions

QCON on other wireless technologies. Our prototype targets today's NSA dual-connectivity deployments because they dominate commercial roll-outs, but the design choices generalize. Multi-connectivity in SA mode with multiple gNBs is also defined in 3GPP standards [11]. The only required change is to adjust PDCP context hand-offs so that 5G BS–5G BS multi-connectivity replaces the current 5G BS-4G BS pairing. Looking forward, early 6G proposals advocate native AI-driven RICs [19, 37]. QCON's transparent app QoE monitoring fits naturally into those blueprints: QoE Monitor becomes one of several "semantic" inputs a 6G RIC will learn from, and QCON's scheduling logic that should infer the app operations, such as adaptive jitter-aware deadline settings, can be enhanced with more sophisticated machine learning

techniques. Our insights apply equally to Wi-Fi multi-link operation [21], which we leave as future work.

QCON with future RTS apps. QCON leverages standard RTP protocol header information, which underpins current RTS apps. For emerging RTS workloads like volumetric video [35,50] or holographic telepresence [39] that mainly focus on en/decoding techniques, the transport protocol remains undecided. Encouragingly, RTP/RTCP continues to be a viable delivery substrate for interactive media: IETF is actively standardizing RTP packet formats for volumetric video [40], and Microsoft has implemented RTP in holographic remoting [2]. This means QCON can continue extracting frame-level insights from standard RTP header fields without modifications. For future RTC apps that may adopt alternative protocols such as TCP or QUIC, we leave this exploration as future work.

QCON to provide new revenue opportunities. Current commercial providers primarily focus on user-centric service models, as evidenced by Verizon's "Enhanced Video Calling" available only on premium plans [42]. However, implementing QCON opens significant new revenue opportunities for 5G operators by offering premium services to mobile RTS app providers whose user QoE is critically determined at the wireless stage. Compared to existing QoS mechanism [14] that may not directly improve the QoE of user, QCON enhances QoE for mobile RTS apps with only wireless-stage optimization. This enables 5G operators to attract both users and app providers seeking guaranteed performance.

Additional deployment considerations are in Appendix D.

9 Related Work

App-aware wireless endpoint. Several literature optimizes wireless endpoints for app-specific requirements. Zhuge [57] reduces tail frame delay for RTS apps by minimizing the control loop through faster congestion detection and notification from wireless endpoints. Tutti [83] and ARMA [86] optimize video analytics app over 5G through tight integration between app and RAN. RAN slicing [16,23] primarily focus on SLAs rather than directly considering the QoE of RTS apps. QCON directly optimizes RTS app QoE while simultaneously considering radio resource efficiency, bridging the gap between app-level metrics and RAN resource utilization.

QoE-driven multi-path in wireless networks. Multi-path solutions have been proposed to enhance app performance by increasing reliability under wireless fluctuations. Converge [26] proposes video-aware multi-path transport over WebRTC. Augur [88] leverages user state stability for multi-path scheduling in real-time video, assuming user state remains relatively stable over short time periods. Other works [55, 87] have proposed QoE-driven multi-path techniques for on-demand video streaming. However, these approaches typically assume users are connected to multiple cellular networks or to cellular networks with Wi-Fi, which are not prevalent in current mobile devices. In contrast, QCON can be deployed with only single-transport in an app and user-transparent manner, making it immediately applicable to existing infrastructure.

10 Conclusion

This paper presented QCON, a QoE-aware 5G multi-connectivity framework to optimize RTS app QoE that works transparently to both apps and users. By bridging the gap between app QoE and the multi-link capabilities, QCON fully leverages the potential of multi-connectivity to maximize the app performance in a radio resource-efficient way. Notably, these improvements are achieved without the significant deployment barriers of the existing multi-path solution (e.g., multiple cellular subscription requirements) for mobile users. Evaluated on open-source radio suite and real-world trace-driven emulation, QCON shows high QoE improvements in WebRTC and commercial cloud gaming services compared to the state-of-the-art multi-path scheduling techniques. Beyond cloud gaming and WebRTC, the core design of QCON can be extended to a variety of latency-sensitive services, bringing the promise of robust, high-quality RTS closer to reality for mobile users everywhere.

Acknowledgments

We thank our shepherd, Sergey Gorinsky, and the anonymous reviewers for their helpful feedback. This work was supported by the National Science Foundation under Grant No. 1908910, in part by the IITP under Grant Nos. 2025-RS-2024-00398157, 2025-RS-2024-00418784, 2025-RS-2020-11201819 and RS-2022-II220420, and in part by ARL under Grant No. W911NF-2020-221. This work was conducted while Mr. Goodsol Lee was visiting the University of Colorado Boulder. S. Bahk, K. Lee and S. Ha are the corresponding authors.

Disclaimer. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Google meet. https://meet.google.com/. Accessed: 2025-04-25.

[2] Holographic remoting version history. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-version-history. Microsoft Learn documentation. Accessed 2025-04-25.

[3] Magma: A modern mobile core network solution. https://magmacore.org/. Accessed: 2025-04-25.

[4] OpenAirInterface: 5G Software Alliance for Democratising Wireless Innovation. https://openairinterface.org/. Accessed: 2025-04-25.

[5] RC4DAT-6G-60 Programmable Attenuator. https://www.minicircuits.com/WebStore/dashboard.html?model=RC4DAT-6G-60. Accessed: 2025-04-25.

[6] Zoom. https://zoom.us/. Accessed: 2025-04-25.

[7] ps(1) — linux manual page. https://man7.org/linux/man-pages/man1/ps.1.html, September 2024. Accessed 2025-04-25.

[8] 3GPP TR 23.107. Quality of Service (QoS) concept and architecture. ver. 18.0.0, 2024.

[9] 3GPP TR 23.207. End-to-end Quality of Service (QoS) concept and architecture. ver. 18.0.0, 2024.

[10] 3GPP TR 36.932. Scenarios and requirements for small cell enhancements for E-UTRA and E-UTRAN. ver. 15.0.0, 2018.

[11] 3GPP TR 37.340. NR; Multi-connectivity; Overall description; Stage-2. ver. 18.3.0, September 2024.

[12] 3GPP TR 38.321. NR; Medium Access Control (MAC) protocol specification. ver. 16.7.0, December 2021.

[13] 3GPP TR 38.322. NR; Radio Link Control (RLC) protocol specification. ver. 18.2.0, December 2024.

[14] 3GPP TS 23.501. System architecture for the 5G System. ver. 19.1.0, September 2024.

[15] ACCUVER. XCAL. https://www.accuver.com/sub/products/view.php?idx=6. Accessed: 2025-04-25.

[16] Arjun Balasingam, Manikanta Kotaru, and Paramvir Bahl. Application-level service assurance with 5G RAN slicing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 841–857, 2024.

[17] Jan Biniok. Tampermonkey. https://www.tampermonkey.net/, 2025. Browser extension for userscript management. Accessed: 2025-04-25.

[18] Boosteroid. Boosteroid cloud gaming. https://boosteroid.com/. Accessed: 2025-04-25.

[19] Christopher G. Brinton, Mung Chiang, Kwang Taik Kim, David J. Love, Michael Beesley, Morris Repeta, John Roese, Per Beming, Erik Ekudden, Clara Li, Geng Wu, Nishant Batra, Amitava Ghosh, Volker Ziegler, Tingfang Ji, Rajat Prakash, and John Smee. Key focus areas and enabling technologies for 6G. *IEEE Communications Magazine*, 63(3):84–91, 2025.

[20] 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*, pages 1–12, 2016.

[21] Marc Carrascosa-Zamacois, Giovanni Geraci, Edward Knightly, and Boris Bellalta. Wi-Fi multi-link operation: An experimental study of latency and throughput. *IEEE/ACM Transactions on Networking*, 32(1):308–322, 2023.

[22] Yongzhou Chen, Ammar Tahir, Francis Y. Yan, and Radhika Mittal. Octopus: In-network content adaptation to control congestion on 5G links. In *2023 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 199–214. IEEE, 2023.

[23] Yongzhou Chen, Ruihao Yao, Haitham Hassanieh, and Radhika Mittal. Channel-aware 5G RAN slicing with customizable schedulers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1767–1782, 2023.

[24] Ruizhi Cheng, Nan Wu, Matteo Varvello, Eugene Chai, Songqing Chen, and Bo Han. A first look at immersive telepresence on Apple Vision Pro. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 555–562, 2024.

[25] Kourosh Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey. *IEEE Transactions on Robotics*, 39(3):1706–1727, 2023.

[26] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Converge: QoE-driven multipath video conferencing over webrtc. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 637–653, 2023.

[27] Linux Kernel Documentation. Tun/tap - virtual point-to-point (tun) and ethernet (tap) devices. https://docs.kernel.org/networking/tuntap.html. Accessed: 2025-01-20.

[28] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. dApps: Distributed applications for real-time inference and control in O-RAN. *IEEE Communications Magazine*, 60(11):52–58, 2022.

[29] eBPF Foundation. eBPF - extended berkeley packet filter. https://ebpf.io/. Accessed: 2024-11-15.

[30] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5G RAN analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.

[31] Xenofon Foukas, Tenzin Samten Ukyab, Bozidar Radunovic, Sylvia Ratnasamy, and Scott Shenker. RANBooster: Democratizing advanced cellular connectivity through fronthaul middleboxes. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 742–757, 2025.

[32] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 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)*, pages 267–282, 2018.

[33] Google. WebRTC official page. https://webrtc.org/. Accessed 2025-04-25.

[34] Google Chrome Team. chrome://webrtc-internals — built-in webrtc diagnostics page. chrome://webrtc-internals/. Built into Chromium-based browsers for real-time WebRTC stats and debugging. Accessed 2025-04-25.

[35] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. Metastream: Live volumetric content capture, creation, delivery, and rendering in real time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[36] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z Morley Mao, Feng Qian, and Zhi-Li Zhang. Vivisecting mobility management in 5G cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 86–100, 2022.

[37] Jakob Hoydis, Fayçal Ait Aoudia, Alvaro Valcarce, and Harish Viswanathan. Toward a 6G AI-native air interface. *IEEE Communications Magazine*, 59(5):76–81, 2021.

[38] Xiangjie Huang, Jiayang Xu, Haiping Wang, Hebin Yu, Sandesh Dhawaskar Sathyanarayana, Shu Shi, and Zili Meng. ACE: Sending burstiness control for high-quality real-time communication. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 1182–1198, 2025.

[39] Keiichi Ihara, Mehrad Faridan, Ayumi Ichikawa, Ikkaku Kawaguchi, and Ryo Suzuki. Holobots: Augmenting holographic telepresence with mobile robots for tangible remote collaboration in mixed reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–12, 2023.

[40] Lauri Ilola and Lukasz Kondrad. RTP Payload Format for Visual Volumetric Video-based Coding (V3C). Internet-Draft draft-ietf-avtcore-rtp-v3c-08, Internet Engineering Task Force, mar 2025. Work in Progress.

[41] Hassan Iqbal, Ayesha Khalid, and Muhammad Shahzad. Dissecting cloud gaming performance with decaf. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(3):1–27, 2021.

[42] Allison Johnson. Verizon is using 5G network slicing to offer better video calling — for a price, 2024. https://www.theverge.com/2024/12/12/24319785/verizon-5G-sa-network-slicing-enhanced-video-calling. Accessed: 2025-04-25.

[43] Rostand A K. Fezeu, Claudio Fiandrino, Eman Ramadan, Jason Carpenter, Lilian Coelho de Freitas, Faaiq Bilal, Wei Ye, Joerg Widmer, Feng Qian, and Zhi-Li Zhang. Unveiling the 5G mid-band landscape: From network deployment to performance and application QoE. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 358–372, 2024.

[44] Seyeon Kim, Kyungmin Bin, Donggyu Yang, Sangtae Ha, Song Chong, and Kyunghan Lee. Entro: Tackling the encoding and networking trade-off in offloaded video analytics. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9115–9123, 2023.

[45] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. EdgeRIC: Empowering real-time intelligent optimization and control in NextG cellular networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1315–1330, 2024.

[46] Goodsol Lee. QCON source code repository. https://github.com/goodsollee/qcon-nsdi26, 2025. Accessed: 2025-08-22.

[47] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. Raven: Improving interactive latency for the connected car. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 557–572, 2018.

[48] Insoo Lee, Jinsung Lee, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Demystifying commercial video conferencing applications. In *Proceedings of the 29th ACM international conference on multimedia*, pages 3583–3591, 2021.

[49] Jinsung Lee, Sungyong Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Hyoyoung Lim, Jihoon Lee, Xiaoqing Zhu, Sangeeta Ramakrishnan, Dirk Grunwald, Kyunghan Lee, et al. Perceive: deep learning-based cellular uplink prediction using real-time scheduling patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 377–390, 2020.

[50] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. Groot: A real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[51] Qianru Li, Zhehui Zhang, Yanbing Liu, Zhaowei Tan, Chunyi Peng, and Songwu Lu. CA++: Enhancing carrier aggregation beyond 5g. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2023.

[52] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and JD Cock. VMAF: The journey continues. *Netflix Technology Blog*, 25(1), 2018.

[53] Linux man-pages project. Linux tc(8) - show/manipulate traffic control settings. https://man7.org/linux/man-pages/man8/tc.8.html, 2024. Accessed: 2025-04-25.

[54] Zhutian Liu, Qing Deng, Zhaowei Tan, Zhiyun Qian, Xinyu Zhang, Ananthram Swami, and Srikanth V Krishnamurthy. M2HO: Mitigating the adverse effects of 5G handovers on TCP. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 1089–1103, 2024.

[55] Gerui Lv, Qinghua Wu, Yanmei Liu, Zhenyu Li, Qingyue Tan, Furong Yang, Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, et al. Chorus: Coordinating mobile multipath scheduling and adaptive video streaming. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 246–262, 2024.

[56] Minzhao Lyu, Sharat Chandra Madanapalli, Arun Vishwanath, and Vijay Sivaraman. Network anatomy and real-time measurement of nvidia geforce now cloud gaming. In *International Conference on Passive and Active Network Measurement*, pages 61–91. Springer, 2024.

[57] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 193–206, 2022.

[58] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 907–926, 2024.

[59] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Enabling passive measurement of zoom performance in production networks. In *Proceedings of the 22nd ACM internet measurement conference*, pages 244–260, 2022.

[60] Microsoft. Xbox cloud gaming. https://www.xbox.com/en-us/play. Accessed: 2025-04-25.

[61] Iain Morris. At&t, all in with ericsson, seems to have shut the door to xapps. *Light Reading*, June 2024. 5 Min Read.

[62] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z Morley Mao. MP-H2: a client-only multipath solution for http/2. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[63] NVIDIA. Geforce now system requirements. https://www.nvidia.com/en-us/geforce-now/system-reqs/#windows-pc, 2025. Accessed: 2025-04-25.

[64] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials*, 25(2):1376–1411, 2023.

[65] Andrey Popov. Arrival of esim is altering how consumers interact with operators. https://www.opensignal.com/2023/07/25/arrival-of-esim-is-altering-how-consumers-interact-with-operators, July 25 2023. Accessed: 2025-04-25.

[66] Carlos Pupiales, Daniela Laselva, Quentin De Coninck, Akshay Jain, and Ilker Demirkol. Multi-connectivity in mobile networks: Challenges and benefits. *IEEE Communications Magazine*, 59(11):116–122, 2021.

[67] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath TCP. In *9th USENIX symposium on networked systems design and implementation (NSDI 12)*, pages 399–412, 2012.

[68] Jaya Rao and Sophie Vrzic. Packet Duplication for URLLC in 5G: Architectural Enhancements and Performance Analysis. *IEEE Network*, 32(2):32–40, 2018.

[69] Ettus Research. USRP B210 SDR kit - dual channel transceiver (70MHz - 6GHz). https://www.ettus.com/all-products/ub210-kit/. Accessed: 2025-03-30.

[70] Claudio Rosa, Klaus Pedersen, Hua Wang, Per-Henrik Michaelsen, Simone Barbera, Esa Malkamäki, Tero Henttonen, and Benoist Sébire. Dual connectivity for lte small cell evolution: Functionality and performance aspects. *IEEE Communications Magazine*, 54(6):137–143, 2016.

[71] William Sentosa, Balakrishnan Chandrasekaran, P Brighten Godfrey, Haitham Hassanieh, and Bruce Maggs. DChannel: Accelerating mobile applications with parallel high-bandwidth and low-latency channels. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 419–436, 2023.

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

[73] sysmocom. sysmocom usim. https://sysmocom.de/products/discontinued/sysmousim/index.html, 2023. Accessed 2025-04-25.

[74] Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. Supporting mobile VR in LTE networks: How close are we? *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):1–31, 2018.

[75] Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, and Songwu Lu. Device-based LTE latency reduction at the application layer. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 471–486, 2021.

[76] Haiping Wang, Zhenhua Yu, Ruixiao Zhang, Siping Tao, Hebin Yu, and Shu Shi. Twinstar: A practical multipath transmission framework for ultra-low latency video delivery. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9234–9242, 2023.

[77] Jialin Wang, Rongkai Shi, Wenxuan Zheng, Weijie Xie, Dominic Kao, and Hai-Ning Liang. Effect of frame rate on user experience, performance, and simulator sickness in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2478–2488, 2023.

[78] Wikipedia. Iperf — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Iperf&oldid=1059070839. Accessed 2024-04-25.

[79] Wikipedia contributors. Real-time Transport Protocol — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Real-time_Transport_Protocol, 2025. [Online; accessed 5-March-2025].

[80] Wikipedia contributors. Secure Real-time Transport Protocol – Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Secure_Real-time_Transport_Protocol, 2025. Accessed: 2025-01-15.

[81] Wikipedia contributors. Zeromq — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/ZeroMQ, 2025. Accessed 2025-04-25.

[82] Yaxiong Xie, Fan Yi, and Kyle Jamieson. Pbe-cc: Congestion control via endpoint-centric, physical-layer bandwidth measurements. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 451–464, 2020.

[83] Dongzhu Xu, Anfu Zhou, Guixian Wang, Huanhuan Zhang, Xiangyu Li, Jialiang Pei, and Huadong Ma. Tutti: coupling 5g ran and mobile edge computing for latency-critical video analytics. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 729–742, 2022.

[84] Fan Yi, Haoran Wan, Kyle Jamieson, Jennifer Rexford, Yaxiong Xie, and Oliver Michel. Athena: Seeing and mitigating wireless impact on video conferencing and beyond. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 103–110, 2024.

[85] Juheon Yi, Sunghyun Choi, and Youngki Lee. EagleEye: Wearable camera-based person identification in crowded urban spaces. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[86] Juheon Yi, Goodsol Lee, Minkyung Jeong, Seokgyeong Shin, Daehyeok Kim, and Youngki Lee. Towards end-to-end latency guarantee in mec live video analytics with app-ran mutual awareness. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*, pages 402–416, 2025.

[87] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 418–432, 2021.

[88] Yuhan Zhou, Tingfeng Wang, Liying Wang, Nian Wen, Rui Han, Jing Wang, Chenglei Wu, Jiafeng Chen, Longwei Jiang, Shibo Wang, et al. AUGUR: Practical mobile multipath transport service for low tail latency in real-time streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1901–1916, 2024.

Appendices

A Commercial 5G Experiments Detail

A.1 Real-world 5G experiments

We collected comprehensive traces on three major 5G operators from February to March 2025, traveling through campus areas, city streets, and highways. Our subscription plans for all three operators are the most premium plan, which is renowned for providing full capabilities of 5G. Table 5 presents the statistics of our bandwidth measurements, collected at 100 ms intervals using Iperf [78] simultaneously on both 5G and 4G Samsung S22 smartphones. Note that Iperf app and there is no control latency from sleep operation (e.g., Wake up latency from DRX mode or RRC idle mode [75]) in the traces.

A.2 Commercial RTS app Measurements

Measuring the app performance of commercial RTS apps presents unique challenges compared to open-source WebRTC, as commercial providers typically do not expose their logging features to users. We discovered that our target commercial cloud gaming platforms—Nvidia GeForceNow, Xbox cloud gaming, and Boosteroid—reveal key performance metrics such as bitrate and frame rates through the standard web-based WebRTC diagnostic tool WEBRTC-INTERNALS [34]. This tool enables extraction and logging of performance data for WebRTC-based streaming apps delivered over the web. To automate data collection, we implemented a Tampermonkey [17] script that captures information from WEBRTC-INTERNALS and transmits it to our web server for storage and analysis. This methodology allowed us to systematically run cloud gaming sessions and extract comprehensive performance metrics during gameplay.

B Fairness of QCON

Figure 21 shows the fairness of QCON. We run an RTS app on one UE while a contending UE loads 2 MB web pages every 15 seconds. We test two scenarios: the contending UE uses either a 4G or 5G link. QCON achieves high RTS performance without compromising the contending UE's web page loading performance. When the contending UE uses 4G, QCON delivers 5.7 s of web loading times comparable to

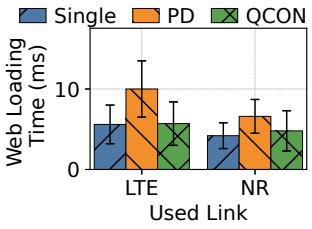


Figure 21: Page loading time of contending UE while serving RTS UE with QCON.

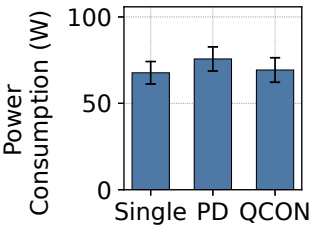


Figure 22: Power consumption of the host device.

SINGLE (5.7 s), which serves the RTS app exclusively over 5G. When the contending UE uses 5G, QCON maintains similarly low web page load times of 4.8 s as SINGLE (4.6 s), even though QCON achieves higher RTS bitrates. PD that achieves similar RTS app performance degrade the page loading time because of its aggressive utilization of both 4G and 5G links.

C Power Consumption of QCON

Figure 22 depicts the power consumption of the host device running QCON. QCON consumes 3.2% more power than SINGLE while achieving higher RTS performance. In contrast, PD consumes over 11% more power to achieve comparable performance to QCON. This is because QCON efficiently utilizes both 4G and 5G links without PD's resource waste.

D Additional Considerations for Real-World Deployment of QCON

In this paper, we implement and evaluate QCON on lab-scale testbeds. To extend QCON to real-world deployments, the following considerations should be addressed.

Handing the wired bottleneck. QCON mainly tackles the wireless bottleneck scenario, which is prevalent in current mobile RTC apps [26]. However, QCON also may encounter a wired bottleneck. In such cases, video frames already experience significant delays over the wired link, and QCON will need to conduct multi-link scheduling based on the actual slack time based on the video frame generation time, rather than the slack time based on a fixed SLA. This may require the cooperation with the apps to notify its accurate slack time to RAN, which we leave as a future work.

Handling the complexity of RIC-based solution. One concern in QCON's real-world deployment is the inherent complexity of RIC solutions, which require RAN modifications to provide observability and controllability to the RIC. This may lead to compatibility issues when RAN and RIC components are from different manufacturers, resulting a barrier to deploying QCON [61]. However, recent research efforts have recognized this issue and are working to enable programmability without modifying RAN source code [30] or to facilitate the transfer of RAN status across different RAN and

Scenario	Case	OpX				OpY				OpZ			
		Mean	P05	P01	P0.1	Mean	P05	P01	P0.1	Mean	P05	P01	P0.1
Campus	4G	76.96	0	0	0	83.62	10.94	0	0	93.38	21.55	0.42	0
	5G	235.34	3.25	0	0	53.63	0	0	0	369.74	166.92	0	0
	4G+5G	310.92	10.33	0	0	138.10	28.54	8.04	0	461.22	225.72	73.02	20.92
City	4G	170.76	46.92	0	0	66.84	1.32	0	0	73.58	5.56	0	0
	5G	420.19	38.16	11.83	0	111.78	4.76	0	0	101.51	32.59	0	0
	4G+5G	593.53	150.32	74.77	34.85	179.31	34.24	8.20	0	173.97	73.73	39.88	7.08
Highway	4G	95.07	2.00	0	0	65.50	7.90	0	0	53.17	0	0	0
	5G	286.34	29.64	0	0	218.15	77.45	23.30	0	111.98	0	0	0
	4G+5G	376.86	58.10	12.12	0	284.17	106.28	47.96	7.32	165.96	26.82	4.85	0

Table 5: Measured bandwidth of 5G, 4G, and simultaneous usage of 5G and 4G in Mbps.

RIC vendors [31]. Aligning with these efforts is expected to accelerate the deployment of RIC-based solutions including QCON.