PAVE: Mitigating Non-Congestive Delay for Seamless Video Calls over NextG Mobile Networks

Goodsol Lee[†], Seyeon Kim[‡], Juheon Yi[†], Junhong Min[¶], Sangtae Ha[¶], Kyunghan Lee[†], Saewoong Bahk[†]
[†]Seoul National University [‡]Korea University [¶]University of Colorado Boulder

***Abstract*—With mobile video calls now ubiquitous, ensuring seamless video-based real-time communications (RTC) remains a critical challenge for 5G operators. Despite abundant 5G bandwidth, video calls frequently experience low quality and unacceptable latency during channel fluctuations—not due to bandwidth limitations or congestion, but because of non-congestive delays in the radio access network (RAN). These delays stem from general-purpose RAN transmission procedures that prioritize radio resource efficiency over application latency through reactive scheduling and static timer-based retransmissions. Existing solutions largely address congestion-induced delays or sacrifice spectral efficiency to mitigate non-congestive delays.**

To address this, we present PAVE, a novel RAN-side solution that breaks the fundamental tradeoff between spectral efficiency and RTC latency. Our key insight is that RTC traffic exhibits distinct characteristics—periodic traffic generation and deadline-driven urgency—that can be leveraged to optimize RTC quality of experience (QoE) without sacrificing efficiency. For practical deployment, PAVE extracts these characteristics at the RAN, enables resource preallocation that goes beyond strictly periodic traffic patterns, and incorporates a selective retransmission and skip mechanism that maintains high spectral efficiency while strategically leveraging application-layer recovery. Implemented in an Open-RAN compliant RAN Intelligent Controller, PAVE improves tail frame rates by 1.8× and reduces video stalls by 94% in real-world evaluations.

***Index Terms*—Real-time communications, Mobile networks, Cross-layer protocol, Open-RAN**

I. Introduction

Video-based real-time communication (RTC) [1], such as FaceTime and Zoom, has emerged as a key application for 5G mobile users. In the first quarter of 2024, over 35% of internet users globally participated in video calls via mobile devices, with younger generations comprising approximately 42% of this group [2]. Similarly, video calls have become ubiquitous in business, with around 80% of professional meetings now relying on video communication [3]. As a result, ensuring seamless RTC experiences has become crucial for 5G operators seeking to attract and retain subscribers. For example, Verizon recently introduced "Enhanced Video Calling," its first customer-facing implementation of network slicing—exclusively for premium customers [4].

However, enabling seamless RTC over 5G remains a significant challenge. To ensure high quality of experience (QoE) despite wireless fluctuations, RTC applications require video frames to sustain high bitrates with consistently low latency under strict deadlines (e.g., 150 ms [5]), even at the tail (e.g., 99th percentile) [6]. Unfortunately, our comprehensive measurements on commercial 5G uplink reveal that WebRTC, the de facto standard for real-time streaming [7], [8], still suffers from frequent stalled video frames, even with high bandwidth.

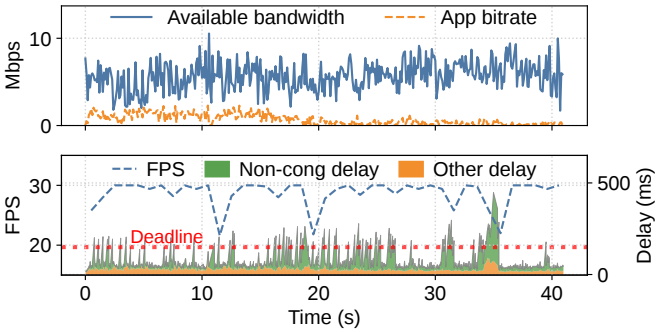


Fig. 1: WebRTC performance over a 5G testbed (§VI). Although bandwidth is sufficient for target bitrates, non-congestive RAN delays frequently cause frame deadline violations, leading to low bitrates and degraded QoE.

To address the degradation of QoE caused by RTC applications, existing works primarily focus on reducing queuing delays caused by network congestion. These approaches either quickly reduce the sending rate at the transport layer using congestion control mechanisms [6], or proactively reserve bandwidth at the link layer through techniques such as network slicing [5]. However, these solutions are designed to react when the available bandwidth falls below the bitrate (i.e., network congestion), and therefore are not effective at mitigating latency issues that persist even when the bandwidth is sufficient.

Surprisingly, contrary to the common assumption that bandwidth drops and resulting queuing are the primary causes of frame stalls [6], our analysis reveals that the dominant contributor to stalled frame delay is non-congestive delay[1] of the radio access network (RAN), accounting for the majority of the total frame delay, as shown in Fig. 1. This delay stems from inefficient RAN transmission procedures that occur regardless of network congestion. While channel fluctuations and successive channel errors exacerbate frame delays, the delay directly attributable to the channel errors themselves accounts for only a small fraction of the overall frame delay.

This work was supported by the National Science Foundation under Grant No. 1908910; by the Institute of Information Communications Technology Planning Evaluation (IITP) under Grant Nos. 2026-2021-002048, 2026-RS-2024-00429088, and 2026-RS-2024-00398157; and by the National Research Foundation of Korea (NRF) under Grant No. RS-2025-24535401. The corresponding authors are S. Ha and S. Bahk.

[1]Non-congestive delay is the delay caused by packets being held due to network elements [9]. In 5G, we consider non-congestive RAN delays occur when the RAN does not allow to transmit packets even though there is sufficient bandwidth (Details in §II-A).

Specifically, we identify two mechanisms in the transmission procedure that exacerbate non-congestive RAN delays: *(1) Request-based uplink scheduling.* To transmit with only the necessary resources in the uplink, the UE (User Equipment) must request resources and wait for a grant, which can take up to 20 ms for each transmission. *(2) Static timer-based retransmission.* 5G employs a two-stage retransmission strategy: immediate retries using channel coding from the initial transmission, followed by delayed retries that adapt to current channel conditions. These delayed retransmissions typically occur by the expiration of a static timer. While this approach helps avoid redundant retransmissions and ensures in-order delivery at the RAN, it can introduce additional delays of up to 50 ms. We observe that these two procedures can occur repeatedly within the transmission of a single frame, adding over 150 ms of delay.

To address the impact of non-congestive delays on RTC QoE, we introduce PAVE, a novel RAN-side mechanism for seamless video calls that mitigates non-congestive RAN delays without compromising radio resource efficiency. A key advantage of PAVE is that it requires only RAN-side modifications and is compatible with all device and application types—similar to widely adopted VoLTE and VoNR (Voice over New Radio) techniques [10]. PAVE targets reductions in non-congestive delay within per-UE transmission procedures, while remaining fully compatible with existing RAN mechanisms that address congestive delay—such as network slicing, which allocates high-level resource budgets for each transmission [5].

To achieve this, PAVE minimizes non-congestive RAN delays for RTC applications by leveraging two fundamental characteristics of real-time video frames: *periodicity* and *urgency*. Rather than relying on low-throughput channel coding to suppress channel errors, our approach eliminates the majority of non-congestive RAN delays without compromising radio resource efficiency through two key innovations: (1) By recognizing the periodic media generation patterns of RTC applications, PAVE proactively allocates uplink resources to reduce scheduling delays. (2) At the same time, it employs urgency-aware retransmission policies that consider frame deadlines when deciding whether and when to trigger retransmissions, thereby avoiding video stalls. However, realizing PAVE poses three key technical challenges:

- **How can RTC application characteristics be inferred in the network?** RTC traffic exhibits periodicity and urgency at the application's frame level, but the RAN only observes events at the link layer. Analyzing RTC behavior purely at the packet level, without frame-level context, can result in inaccurate interpretations. PAVE addresses this by leveraging standard RTC protocol metadata embedded in each RTC packet, enabling the RAN to access frame-level information directly.
- **How can we reduce non-congestive RAN delays for RTC applications without radio resource efficiency degradations?** RTC traffic characteristics can indicate when scheduling or retransmissions are needed, but relying solely on this information can compromise radio resource efficiency. PAVE combines RTC application characteristics with link-layer information at the RAN to preallocate resources or trigger retransmissions, bypassing standard scheduling procedures to ensure low delay without sacrificing resource utilization.
- **How can we protect the QoE of RTC applications under unpredictable wireless fluctuations?** While PAVE reduces non-congestive RAN delays, this alone may not improve the QoE of RTC applications. Retransmitting delayed RTC frames after their deadlines only wastes radio resources. To address this, PAVE selectively decides whether to retransmit or skip RTC frames based on their deadlines. This allows RTC applications to maintain quality of experience (QoE), as they can often recover entire frames using an application-layer recovery mechanism, even when only a subset of packets is received [11].

We implement PAVE as an application within the OpenRAN RIC (RAN Intelligent Controller) architecture [12], [13], enabling seamless deployment while maintaining compatibility with existing RAN mechanisms. Our real-world evaluation demonstrates that PAVE enhances tail frame rates by 1.8× with 94% reduced video stalls in real-world evaluations.

In summary, our contributions are threefold:

- We systematically analyze non-congestive delays in 5G RAN and their detrimental impact on RTC applications.
- We propose PAVE, a novel RAN-side mechanism that mitigates non-congestive RAN delays by leveraging application-level awareness and link-layer insights.
- We present an OpenRAN-compliant implementation of PAVE and demonstrate, through real-world evaluation, that it enables seamless mobile RTC.

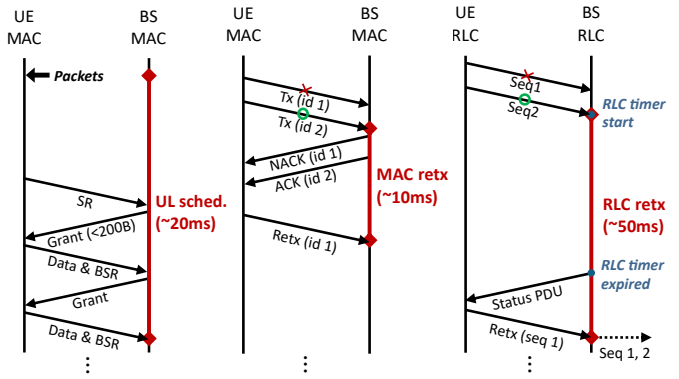
II. Backgrounds

This section details the 5G uplink transmission procedure and provides background on WebRTC, the de facto standard for RTC applications.

A. 5G Data Transmission in the Uplink

5G UEs transmit data through 3GPP-defined procedures that incorporate radio resource scheduling and error recovery mechanisms [14], [15]. This subsection focuses on the uplink path, which we found to be a major performance bottleneck for RTC applications such as video calls [7].

Uplink scheduling procedure. 5G uses request-based scheduling to allocate radio resources according to UE demand, as illustrated in Fig. 2a. When the UE's transmission buffer becomes non-empty, it sends a Scheduling Request (SR) to the base station (BS) in periodic slots (e.g., every 10 ms). The BS then grants a small uplink resource for the UE to send a Buffer Status Report (BSR), which provides information about the size of its buffer. Based on the BSR, the BS allocates additional resources for data transmission. Each BSR-based scheduling round typically takes around 5 ms, resulting in potential scheduling delays ranging from 5 to 20 ms before the UE can transmit its data.



(a) Uplink scheduling. (b) MAC tx/retx. (c) RLC retx.

Fig. 2: Packet transmission procedure in 5G RAN, which can incur non-congestive RAN delays.

Transmission and retransmission procedure. In each slot (0.5 ms in sub-6GHz 5G), the BS allocates transport block size (TBS) to each UE based on modulation and coding schemes (MCS) and resource blocks (RB). The MCS determines how bits are encoded across frequency resources and is selected dynamically via link adaptation, based on current channel conditions. To maximize spectral efficiency, the BS typically selects a high MCS with target block error rates (BLERs) of 10% [16]. When transmissions fail, 5G employs a two-stage retransmission strategy, as shown in Figs. 2b and 2c:

- **MAC layer.** The Hybrid Automatic Repeat Request (HARQ) provides error recovery [14] in the MAC layer. HARQ retransmissions reuse the same resource allocation and redundancy from the original transmission attempt. These occur in 10 ms cycles, based on ACK/NACK feedback, and continue until a maximum retry limit is reached (e.g., 4 attempts [17]).
- **RLC layer.** When HARQ fails, the Radio Link Control (RLC) layer applies Automatic Repeat Request (ARQ) [18]. RLC retransmissions are triggered when the receiver sends a status Protocol Data Unit (PDU) containing ACK/NACK information for specific sequence numbers. In cases of successive channel errors, RLC detects out-of-order packets and initiates a reordering timer (e.g., t-Reassembly = 50 ms [15]) before generating a status PDU to request retransmission. Once triggered, retransmission packets are placed at the front of the transmission buffer with highest priority, while already-arrived packets at the BS must wait in the reordering buffer until the missing packets are recovered.

Delays due to these procedures occur regardless of available bandwidth and are considered non-congestive RAN delays.

B. WebRTC Overview

WebRTC is the de facto standard for real-time video streaming in applications such as video calls. In WebRTC, the sender captures and encodes video frames, then transmits them as packets over the network. The receiver collects the packets for each frame, decodes them, and renders the video in real time.

RTP/RTCP packets. The WebRTC sender packetizes media content and delivers it over UDP with low latency using RTP (Real-time Transport Protocol) packets [19]. Upon receiving RTP packets, the receiver periodically sends RTP Control Protocol (RTCP) packets containing feedback such as packet loss ratio, jitter, and round-trip delay, enabling the sender to adapt to changing network conditions.

Google Congestion Control (GCC) [20]. WebRTC uses GCC as its congestion control algorithm (CCA) to support real-time media over unpredictable networks. It dynamically adapts the sending bitrate to match available bandwidth and avoid congestion. GCC relies on receiver-side delay measurements and bandwidth estimation, which are communicated to the sender via RTCP feedback. For example, if the trend in packet delay increases, GCC reacts conservatively by reducing the sending rate to prevent network overload.

Experiment	Avg. 5G BW	Avg. bitrate	p99/p99.9 frame delay	Stall rate >150 ms
Comm. (OpX)	61.3 Mbps	3.6 Mbps	253/364 ms	5.3%
Comm. (OpY)	53.7 Mbps	3.7 Mbps	212/341 ms	4.7%
srsRAN (OpX)	25.1 Mbps	2.5 Mbps	217/331 ms	6.2%
srsRAN (OpY)	23.5 Mbps	2.4 Mbps	194/312 ms	5.1%

TABLE I: WebRTC performance over commercial 5G and the srsRAN testbed with identical 5G settings for walking scenarios. Despite average bandwidth over 24 Mbps in all experiments, WebRTC shows much lower bitrates than its 5 Mbps max and suffers from high frame delays and stall rates.

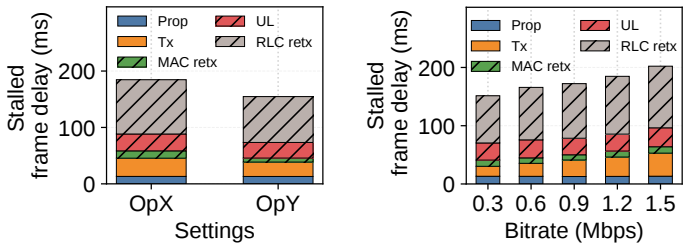
III. Unveiling RTC App Performance over 5G

This section presents the performance of RTC applications over 5G networks and demonstrates how non-congestive RAN delays can degrade their quality.

A. Poor Performance of RTC applications on 5G

We measure WebRTC performance—the de facto standard for RTC applications—on commercial 5G networks and the srsRAN testbed. Experiments are conducted using a Galaxy S22 smartphone transmitting 1080p@30fps video at a maximum bitrate of 5 Mbps [21] to an AWS server with a minimum RTT of 25 ms, over commercial 5G standalone networks from two operators (OpX and OpY), during 15-minute walking scenarios. To complement these commercial network measurements, we conduct srsRAN testbed experiments using identical MAC/RLC configurations extracted from 5G control messages, dissected with the XCAL tool [22], and emulated channels based on SINR traces collected from the commercial networks.

Table I shows average uplink bandwidth measured by iPerf, bitrate, tail frame delay (p99/p99.9), and stall rates (frame delay $>$ 150 ms) of WebRTC applications across different RAN environments. Despite WebRTC's average bitrate demand being only 6%$\sim$10% of the average 5G bandwidth, it experiences high tail frame delays of up to 364 ms. This causes frequent video stalls up to 7.5%, resulting in a delayed video frame every 0.4 second on average. Additionally, WebRTC reduces bitrates in response to high latency through its latency-sensitive GCC algorithm to maintain low latency, resulting in low video



(a) Different configurations. (b) Fixed low bitrates.

Fig. 3: Delay breakdown for stalled video frames on srsRAN testbeds. Most delays stem from non-congestive RAN delays including UL (Uplink scheduling), RLC retx, and MAC retx.

quality. As highlighted in prior work [17], high tail frame delays cause frequent bitrate degradation.

Observation #1. RTC applications exhibit poor performance due to high tail frame delays, even when sufficient 5G bandwidth is available.

B. Non-Congestive RAN Delays on Poor RTC Performance

We next analyze the root cause of long tail frame delays in WebRTC over 5G by classifying the delay components within the 5G RAN (detailed measurement methodology is described in §VI). Fig. 3a presents the delay breakdown of stalled video frames in the srsRAN testbed. In both the OpX and OpY configurations, 75% and 77% of the total delay, respectively, originate from non-congestive RAN delays. In contrast, transmission delay, which includes all resource-allocated transmission attempts for both successful and failed due to channel errors, takes only 16% of the overall delays. Notably, the channel error itself is not the major problem, since it only takes 8% of the total delay, and the associated MAC-layer retransmission delay contributes only a small fraction of the non-congestive delay. Since propagation delay (Prop) is unavoidable, the majority of non-congestive RAN delay, 94% in the OpY setting, is attributed to the combination of uplink scheduling delay [23] and RLC-layer retransmission delay [18]. These delays come from multiple scheduling and RLC retransmission procedures during successive channel errors occurrences, which cause unreported BSRs or out-of-order packet reordering delays.

Such non-congestive RAN delays cannot be mitigated by simply reducing application bitrates to suppress congestion-induced delays. Fig. 3b illustrates the delay breakdown of stalled frames under fixed low bitrates ranging from 0.3 Mbps to 1.5 Mbps in OpX configurations. While decreasing the bitrate may reduce transmission delay, frame stalls still persist due to non-congestive RAN delays. These results indicate that existing approaches, which aim to reduce latency by adapting sending rates to available bandwidth [6] or by reserving link bandwidth to match application bitrates [5], are ineffective in addressing this dominant source of delay.

Observation #2. These high tail frame delays are not caused by network congestion, but instead stem from non-congestive delays inherent to RAN transmission procedures.

C. Approach

To address this issue, it is essential to reduce non-congestive RAN delays to meet the stringent requirements of RTC applications. While conservative MCS selection can mitigate such delays by lowering channel error rates, our srsRAN testbed experiments reveal that this approach severely degrades spectral efficiency. For example, reducing the target BLER from 10% to 0.1% results in a 42% drop in throughput in the driving scenario (Table II). Instead, we focus on optimizing the transmission procedure without sacrificing radio resource efficiency. Specifically, we leverage two key characteristics of RTC applications to design efficient transmission mechanisms:

1) *Periodicity-based proactive grant:* RTC applications typically generate media traffic with predictable periodicity (e.g., 30 FPS video results in frame arrivals 33 ms apart). By proactively allocating small uplink resource blocks based on this periodicity, we can bypass the SR-grant procedure and eliminate most scheduling delays without wasting resources.
2) *Urgency-aware RLC retransmissions:* Instead of relying on a static timer, we selectively trigger or skip retransmissions based on the application urgency, enabling RLC to meet RTC application deadlines more effectively.

IV. PAVE Design

This section presents PAVE, a RAN-side–only mechanism that enhances video calls over 5G while remaining complete transparency to RTC applications.

A. Design Challenges

C1. Inferring RTC application characteristics in the RAN. Since RTC traffic is generated at the frame level [8], it is necessary to infer frame-level information at the RAN to determine the periodicity and urgency of traffic. However, the RAN only observes link-layer events, making such inference challenging. PAVE must infer RTC characteristics without requiring application-layer assistance.

C2. Optimizing both non-congestive RAN delays and radio resource efficiency. While RTC traffic characteristics can indicate when to schedule transmissions or perform retransmissions, periodic application arrival information may be delayed. Moreover, triggering retransmissions solely based on the urgency of current traffic can lead to poor resource efficiency if done at suboptimal times. In our experiment with srsRAN testbed (§VII), the spectral efficiency immediately after the out-of-order event is only 24% of the spectral efficiency 100 ms after the event. PAVE must jointly optimize non-congestive RAN delays reduction and radio resource efficiency.

C3. Protecting QoE of RTC applications under unpredictable channel conditions. Reducing non-congestive RAN delays can certainly help improve QoE, but doing so after application deadlines only wastes radio resources and worsens the situation. Moreover, such decisions must be guided by application-level QoE, meaning that PAVE must intelligently address these scenarios. For instance, if a frame's deadline has passed, a QoE-aware decision may be to skip its recovery and

allow the application to handle the resulting loss using existing application-layer recovery mechanisms [11].

B. Key Design Ideas

RTC characteristics from standard protocol metadata. PAVE leverages metadata in the standard RTP header [19], which is widely adopted by RTC applications. RTP headers include unencrypted frame membership information for each packet. This metadata was originally intended to help receivers avoid decoding incomplete frames. Thus, it poses no security risks and remains in plaintext, even in secure RTP (SRTP) implementations which encrypt only the media payload [24]. By extracting this frame-level information, PAVE identifies frame periodicity and computes per-frame slack time (i.e., the time remaining until a frame's deadline) at the RAN.

Cross-layer information-assisted RAN procedure optimization. PAVE leverages both application-level and RAN-level information to reduce non-congestive RAN delay in transmission procedures. Specifically, it minimizes scheduling and retransmission delays in a radio resource-efficient manner by using cross-layer insights.

- *RAN-assisted RTC traffic scheduling.* It estimates periodic application traffic arrival using metadata from the RTC header and allocates resources for both data transmission and retransmissions.
- *Spectrally efficient retransmission within deadline.* It selectively triggers retransmissions that maximizes spectral efficiency before the deadline.

QoE-driven selective retransmission and frame skipping. To handle out-of-order packets efficiently over fluctuating channels, PAVE adopts two strategies based on slack time information in each of the following two cases:

- *Deadline approaching.* PAVE calculates the expected retransmission time for out-of-order packets and triggers retransmission if it can be completed within the available slack time before the deadline.
- *Deadline passed.* PAVE skips retransmission and flushes reordering queues once the slack time is exhausted, allowing application-level recovery using partial frames [11].

These strategies maximize spectral efficiency under the deadline while ensuring timely frame delivery.

V. PAVE System

This section provides an overview of PAVE and details its core modules. Fig.4 illustrates PAVE as an application running on an Open-RAN-compliant real-time RAN Intelligent Controller (RIC) [12], [13] that manages the RAN. PAVE implements three core modules: The metadata-based RTC profiler (§V-A) extracts frame-level periodicity and urgency from RTP headers (❶ in Fig.4), and hands it over to two control modules (❷). The cross-layer resource preallocator (§V-B) proactively grants radio resources by predicting traffic patterns using both application and RAN states, and indicates BSR for preallocation from RIC (❸). The urgency-aware RLC manager (§V-C) triggers retransmissions in good channel conditions

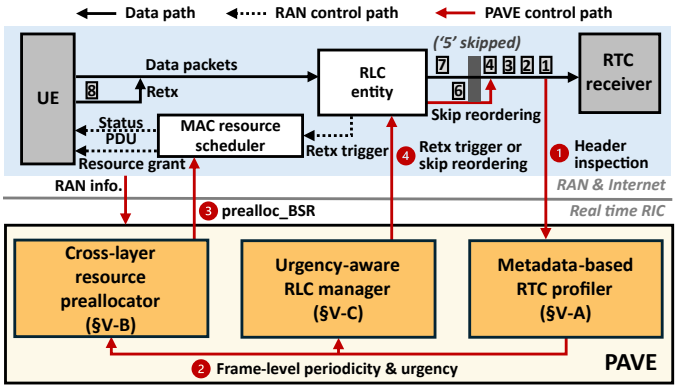


Fig. 4: System overview of PAVE.

or when the deadline is approaching, and skips reordering if the deadline has been passed (❹). While we focus on uplink transmission, PAVE's design principles are also applicable to downlink scenarios (§VIII).

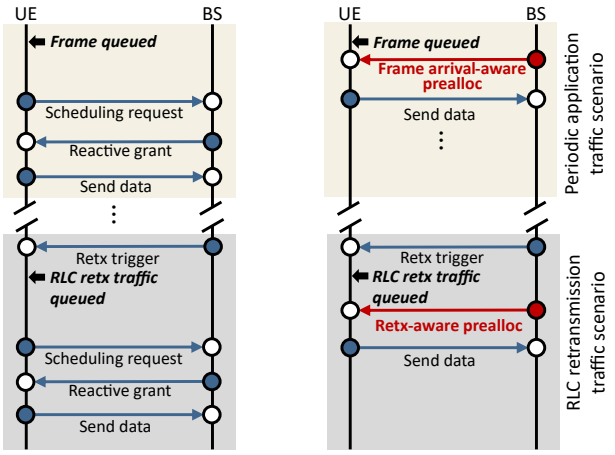
A. Metadata-based RTC Profiler

The metadata-based RTC profiler serves as the foundation of PAVE by extracting frame-level characteristics from standard RTP headers, enabling the system to understand application semantics without requiring modifications to the application.

RTP header inspection. The profiler inspects RTP packet headers to extract frame-level information that characterizes RTC traffic. We filter RTP traffic from RAN egress packets using 5-tuple information (IP addresses and ports) in the IP/UDP headers. From the filtered RTP packets, we extract two key header fields: *1) Timestamp*– represents the relative capture time of a frame; all packets belonging to the same frame share the same timestamp. *2) Marker bit*– indicates the end of a frame by marking the last packet of each frame. These can be adopted in both video and audio streams.

RTC characteristics extraction. The timestamp and marker bit allow us to analyze the periodicity of each frame, while the marker bit enables accurate tracking of frame completion.

- *Periodicity estimation.* Frame periodicity is estimated by measuring timestamp differences between consecutive completed frames. Because RTP timestamps use codec-specific clock rates (e.g., 90 kHz for H.264 video), the profiler converts these values to millisecond-level timing. To account for encoding variability and dynamic network conditions, periodicity estimates are updated every second using a sliding window approach that balances responsiveness and stability.
- *Urgency determination.* Frame urgency is determined by combining marker bit information with application SLA (Service Level Agreements) requirements [5], as conveyed by the core network through standard 5G QoS mechanisms [5]. For uplink traffic, where frames originate at the UE, the profiler estimates frame departure time by analyzing the relationship between the frame generation time (derived from RTP timestamps) and the first packet arrival time observed at the RAN. This correlation enables accurate deadline tracking without requiring modifications on the UE side.



(a) Default 3GPP. (b) PAVE.

Fig. 5: PAVE reduces SR-related procedures, cutting up to 15 ms from the maximum 20 ms per-packet scheduling delay.

The metadata-based RTC profiler then forwards these RTC characteristics to the control modules of PAVE.

B. Cross-layer Resource Preallocator

The cross-layer preallocator minimizes scheduling delays by preemptively allocating radio resources based on traffic-informed buffer estimates.

When to preallocate radio resources? To fully eliminate scheduling delays, radio resources should be preallocated as soon as a packet enters the UE buffer. To achieve this, we estimate two types of traffic arrivals using cross-layer information. Fig. 5 depicts how our approach effectively reduces the scheduling delays in the following two procedures:

- *Periodic application traffic.* While RTC applications typically generate periodic traffic, encoding jitter introduces variation in arrival times. To address periodicity drift, we implement an adaptive prediction mechanism: if a frame does not arrive within the expected window, we extend the prediction interval by the typical BSR delay until the traffic is detected. Once detected, we reset the baseline for future predictions.
- *Aperiodic RLC retransmission traffic.* Unlike application traffic, retransmission timing can be precisely predicted because the RAN tracks retransmission triggers via status PDUs (Retx trigger). We estimate the arrival time of retransmission traffic based on the delivery time of the status PDU to the UE.

Using these estimations, we proactively allocate small radio resources in a timely manner, reducing SR-induced delays.

Standard-compliant implementation. 3GPP standards define resource preallocation techniques, such as semi-persistent scheduling used in VoLTE [10], which predefine both the periodicity and resource grant size. However, such approaches can lead to resource wastage by reserving channel bandwidth regardless of actual RTC traffic demand. Moreover, modifying the RAN scheduler to support dynamic preallocation may conflict with existing standards. To address this, PAVE leverages the BSR defined by 3GPP: the RIC generates a BSR when traffic arrival is expected and transmits it to the scheduler. This approach remains compatible with 3GPP standards without modifications to the MAC scheduler. Following the implementation in the open-source radio suite [25], we set the size of the preallocated BSR to 200 B, which is the grant size from SR.

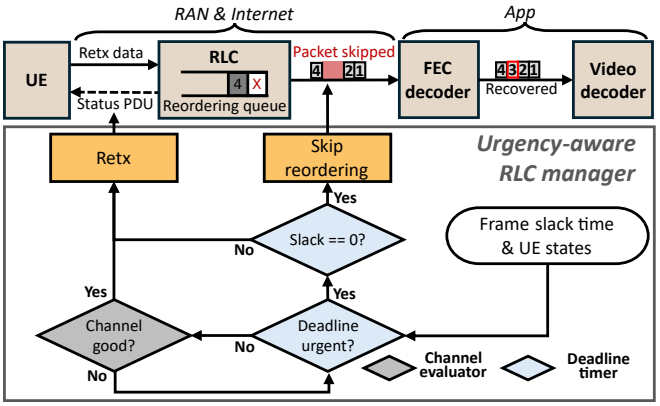


Fig. 6: PAVE triggers retransmission under good channel conditions with sufficient time before the deadline; otherwise, it skips frames that do not contribute to QoE and relies on application-level recovery to avoid deadline violations.

C. Urgency-aware RLC Manager

To meet application requirements while maximizing spectral efficiency, the urgency-aware RLC manager intelligently decides whether to trigger retransmissions for out-of-order packets when conditions are favorable, or to skip frame recovery, delegating loss handling to the application via existing mechanisms such as forward error correction (FEC).

When to trigger retransmission or skip frames? Fig. 6 illustrates how PAVE manages out-of-order packets by considering both application urgency and channel conditions. Initially, the system prioritizes spectral efficiency when sufficient time remains before the application deadline. This deadline is determined by evaluating the urgency of the oldest yet to deliver frame as identified by the marker bit.

- *Channel-aware retransmission.* We compare the current SINR (Signal-to-Interference-plus-Noise Ratio) with that from a recent window (e.g., 50 ms before the out-of-order event) before out-of-order occurrence, which indicates the channel quality before degradations. Retransmission is triggered if the current spectral efficiency matches or exceeds the previous value, indicating improved channel conditions. For urgent frames nearing their deadlines, retransmission is triggered unconditionally if the expected transmission time exceeds the remaining slack time. Specifically, we estimate UE i's expected transmission time T_i for retransmission bytes $ReTx_i$, which can be achieved by multiplication of transmission unit of RLC layer and the number of out-of-order sequences, using the available bandwidth BW_i:

$$T_i = \frac{ReTx_i}{BW_i},\ BW_i = SE_i \cdot \left(R_i + \frac{R_{\max} - \sum_{j \neq i} R_j}{N}\right), \quad (1)$$

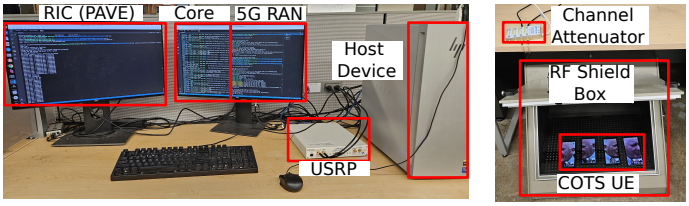


Fig. 7: Open-source 5G radio suite-based Open RAN testbed.

where SE_i is the measured spectral efficiency for UE i, R_i is the number of resource blocks allocated to the user, and N is the total number of active UEs [26]. After triggering retransmission, we set a 20 ms prohibit timer to prevent excessive retransmissions, waiting up to two MAC retransmission attempts. This timer is halted when new out-of-order packets are detected through retransmissions.

- *Urgency-aware frame skipping.* When slack time reaches zero and retransmission cannot meet the deadline, out-of-order packets are flushed to enable application-level recovery. Packets are flushed frame-by-frame by checking timestamps until a frame with sufficient slack time is found.

Standard-compliant implementation. For retransmission control, PAVE triggers retransmissions by sending RIC signals that prompt the generation of status PDUs. While standard RLC timers remain active, PAVE's RTC-aware transmission pipeline prioritizes these RIC signals to accelerate retransmission when needed. To skip frames that do not contribute to QoE, we leverage the existing packet discard functionality of RLC Unacknowledged Mode (UM), which discards out-of-order packets once the reassembly timer expires. PAVE sends RIC signals to trigger packet flushing, continuing this process until non-urgent frames are detected based on frame timestamps.

VI. Implementation

We implement PAVE on a real-world testbed using the open-source 5G radio suite srsRAN Project [25]. It includes ∼10k lines of new/modified C++ code[2] and is built as a μApp on EdgeRIC [13], an Open-RAN-compliant RIC framework for millisecond-scale control. The metadata-based RTC profiler uses eBPF to capture RTP headers with minimal overhead [27], filtering outgoing RTP packets directly at the network interface without explicit packet filtering within the RAN. We extend srsRAN with new RAN functions for cross-layer resource preallocation and urgency-aware RLC management.

Testbed. Our end-to-end Open-RAN testbed consists of srsRAN 5G [25] as RAN software and Open5GS as core network [28]. Hardware includes a NI USRP 2943R radio unit and a Linux desktop with i7-12700 CPU and 32 GB RAM. We use up to 4 commercial off-the-shelf smartphones (three Pixel 6a, one Pixel 7 Pro) with programmable sysmoISIM-SJA2 SIM cards [29], sending traffic to AWS servers with 25 ms minimum RTT. The 5G BS operates in standalone mode with 40 MHz bandwidth (n78 band), numerology 1 (0.5 ms slot time), single MIMO layer, and 5DDDSU TDD duplex mode (2.5 ms uplink periodicity) [30], [31]. Experiments use an RF shield box with Mini-Circuits programmable channel attenuator [32] to emulate real-world SINR traces in mobile scenarios of walking and driving as described in Table II. Unless otherwise stated, our experiments are conducted in the walking scenario.

[2]https://github.com/goodsollee/pave-infocom26

TABLE II: Experimental scenarios.

Scenario	Content	Env	SINR	Bandwidth
Walking	[34]	Campus	20.5±7.9 dB	23.5±6.8 Mbps
Driving	[35]	Highway	17.3±9.2 dB	19.7±8.2 Mbps

Delay analysis methodology. We conduct frame-level delay analysis by mapping each 5G slot to delay components based on link events. Using PTP-synchronized network elements (UE, BS, and server) [33], we measure frame departure/arrival time on each element and classify delay into [9]: propagation delay (frame travel time to server after leaving RAN), transmission delay (sum of resource allocation slots), and non-congestive delays comprising: 1) *Uplink scheduling delay:* slots when UE BSR is unreported to BS scheduler. 2) *MAC retransmission delay:* idle slots waiting for HARQ retransmission. 3) *RLC retransmission delay:* slots when RLC reordering queue waits for out-of-order packets.

VII. Evaluation

We evaluate PAVE with the following aspects:

- **Real-world performance improvements.** Our real-world evaluation shows that PAVE achieves 1.8× improvement in tail frame rates by reducing video stall rates by 94% compared to the state-of-the-art technique.
- **Compatibility with commercial applications.** PAVE is compatible with commercial video call applications, and shows up to 1.4× improvements in tail frame rates.
- **Component effectiveness.** We evaluate the performance of each module in PAVE, demonstrating its ability to effectively handle non-congestive RAN delays.
- **System scalability.** WebRTC with PAVE maintains fairness with other applications while achieving acceptable CPU and memory usage as the number of UE devices grows.

A. Experimental Setup

We validate the performance of PAVE in the following scenarios. Each experiment lasts for one hour.

Applications. We mainly evaluate PAVE using WebRTC [1], as it serves as the de facto standard for the majority of video call applications. WebRTC supports a maximum bitrate of 5 Mbps for transmitting 1080p@30fps video. We evaluate PAVE with various commercial video call applications [36], including Webex [37], GoToMeeting [38], and Google Meet [39].

Baselines. We compare PAVE with the following existing 5G RAN solutions designed for latency-sensitive applications:

- Default represents the existing 5G systems (OpY).
- Zhuge is the last-mile router (e.g., BS) solution [6] that immediately reduces the bitrates when it detects the bandwidth drop to reduce the congestive delay.
- QoS is used in VoLTE or VoNR to reduce the scheduling delay of small traffic. This preallocates small radio resources

Technique	Walking					Driving				
	p99/p99.9 delay	Stall	p01/p01.1 FPS	Bitrate	VMAF	p99/p99.9 delay	Stall	p01/p01.1 FPS	Bitrate	VMAF
Default	194/312 ms	5.1%	24/15	2.4±0.3 Mbps	68.1±9.2	232/362 ms	6.4%	21/13	2.4±0.4 Mbps	74.5±5.3
Zhuge	175/251 ms	4.2%	23/15	1.7±0.3 Mbps	58.3±9.0	199/275 ms	5.6%	21/9	1.1±0.1 Mbps	57.7±7.6
URLLC	167/253 ms	2.1%	24/20	2.4±0.2 Mbps	68.0±9.1	163/296 ms	2.6%	23/18	2.5±0.4 Mbps	76.5±8.4
QoS	162/281 ms	5.1%	23/17	3.2±0.5 Mbps	73.3±7.4	211/315 ms	5.9%	23/12	2.8±0.4 Mbps	78.9±8.1
PAVE	**102/167 ms**	**0.3%**	**28/25**	**3.8±0.5 Mbps**	**76.2±8.7**	**132/174 ms**	**0.6%**	**27/24**	**3.6±0.6 Mbps**	**82.2±7.9**

TABLE III: Application performance across two mobility scenarios.

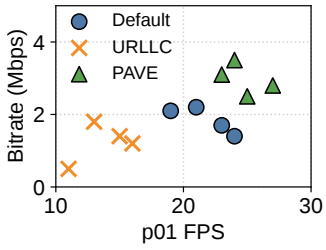


Fig. 8: Multi-user scenarios.

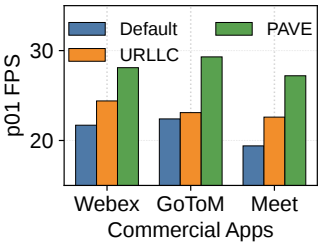


Fig. 9: Commercial apps.

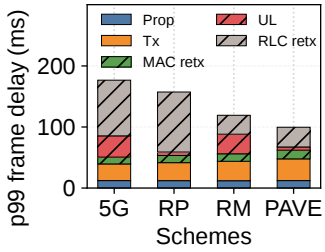


Fig. 10: Ablation studies with tail frame delay breakdown.

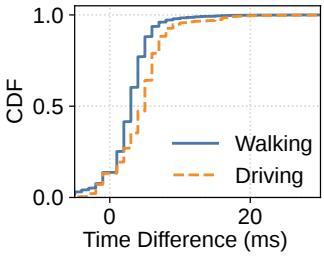


Fig. 11: Frame departure time estimation accuracy.

as in semi-persistent scheduling [10]. We adopt this scheme to preallocate radio resources for every slot as recent private 5G configurations in [17].

- URLLC is a link-layer technique to deliver small traffic with low latency [40]. We set the short RLC retransmission timer (i.e., t-reassembly [15]) as 5 ms and target BLER of link adaptation techniques as 0.1%.

Metrics. We evaluate PAVE with the following five metrics: 1) *Frame delay* is end-to-end delay that should keep deadline (150 ms in video calls [5]). 2) *Frame stalls* is the ratio of stalled video frames that violated the deadline. 3) *FPS* is the frames per second that directly affects the smoothness of the video playback. 4) *Bitrate* is the encoding rate based on bandwidth estimation of RTC CCA, which directly affects video quality. 5) *VMAF* is a video quality metric from Netflix, designed to accurately assess video quality in a way that aligns with human perceptions, with a score between 0 and 100 [41].

B. Real-World Performance Improvements

We first highlight the application performance improvements of PAVE in varying scenarios.

Performance under varying mobilities. Table III shows PAVE significantly outperforms baseline approaches across various mobility scenarios, achieving higher VMAF and bitrate while maintaining lower stall ratios. In driving scenarios, PAVE reduces p99 and p99.9 tail frame delays by 43% and 52%, respectively, compared to DEFAULT. In walking scenarios, it achieves a 94% reduction in stall rates (down to 0.3%) and delivers 1.8× higher tail FPS with a 1.5× higher average bitrate (3.6 Mbps) than DEFAULT. While URLLC improves tail delays, it underperforms relative to PAVE due to congestion from reduced radio efficiency. ZHUGE shows the lowest bitrates, as it frequently lowers transmission rates in response to non-congestive delays without effectively reducing frame delay.

Performance with multiple UEs. In Fig. 8, we evaluate performance under a 4-UE setup. PAVE achieves higher tail FPS and 1.6× higher average bitrate than DEFAULT by reducing non-congestive delays without sacrificing radio resource efficiency. In contrast, URLLC performs worse than DEFAULT due to increased queuing delays caused by reduced radio efficiency and bandwidth as the number of contending UEs increases.

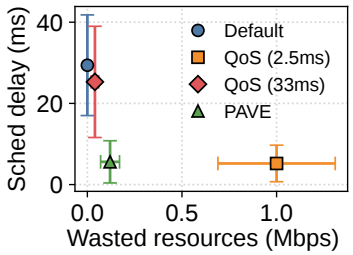


Fig. 12: Microbenchmark on resource preallocation.

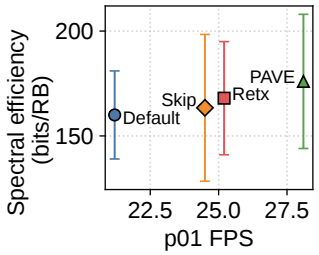


Fig. 13: Microbenchmark on RLC manager.

Performance of commercial applications. Fig. 9 evaluates PAVE with commercial video conferencing applications (Webex, GoToMeeting (GoToM), Google Meet (Meet)) with 720p@30fps. Using analysis from WEBRTC-INTERNALS [42], which provides overall statistics for WebRTC-based sessions in the browser, we find that PAVE up to 1.4× higher p01 FPS than URLLC in Webex, GoToMeeting, and Google Meet, respectively, demonstrating full compatibility with the RTP.

C. Microbenchmarks

Ablation study. We compare three configurations: RP (resource preallocator only), RM (RLC manager only), and the full PAVE system. Fig. 10 shows that each component effectively reduces delay: RP lowers uplink scheduling delay by 85%, and RM reduces RLC retransmission delay by 66%, enabling PAVE to achieve a 38% reduction in overall p99 frame delay compared to default 5G. PAVE increases the transmission delay of tail frame, but this is due to its higher bitrate.

Metadata-based RTC profiler accuracy. Fig. 11 shows high accuracy in frame departure time estimation, with 98.5% and

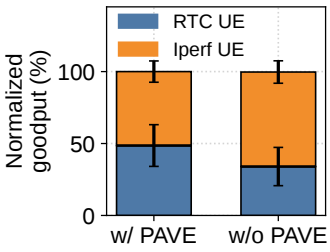


Fig. 14: Fairness of PAVE.

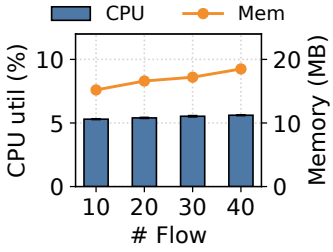


Fig. 15: Overhead of PAVE.

95.7% of samples falling within a 10 ms error margin under walking and driving conditions, respectively, enabling precise determination of frame-level periodicity and urgency.

Resource preallocation technique. As shown in Fig. 12, PAVE achieves a scheduling delay of 5.2 ms for frame delays above p99, which is comparable to QoS(2.5 ms) that preallocates radio resources every slot. However, PAVE significantly reduces radio resource wastage to only 0.12 Mbps by selectively minimizing SR-induced procedures. This represents a substantial improvement over naive periodic allocation (QoS (33 ms)).

Selective retransmission and frame skipping mechanism. We conduct an ablation study comparing retransmission-only and skip-only variants against the full PAVE system. In Fig. 13, PAVE achieves 1.1× higher spectral efficiency than the default 5G, while improving tail FPS. Note that this enhancement is due to the efficient RLC retransmission of PAVE that achieves 1.7× higher spectral efficiency in RLC retransmission traffic.

D. PAVE *Deep Dive*

Fairness with other traffic. We evaluate the fairness of PAVE using a single TCP Cubic flow [43] and a single WebRTC session with no bitrate limitations to assess how it utilizes the available bandwidth. Fig. 14 shows that PAVE maintains fairness while utilizing more bandwidth: RTC traffic with PAVE captures 48.6% of the total bandwidth, compared to 34% without PAVE. Cross-traffic experiments further show that WebRTC over PAVE can effectively compete with high-buffer TCP flows by reacting selectively to queuing delays.

Overheads. Finally, we evaluate the system-level overheads of PAVE. Fig. 15 evaluates system overhead using dummy 5 Mbps RTP flows. PAVE shows minimal scaling overhead: CPU utilization remains stable at 5.3-5.4% (10-40 flows), while memory usage increases modestly from 15.2 MB to 18.5 MB, demonstrating efficient scalability.

VIII. Discussions

Applicability to downlink. While PAVE focuses on video calls in the uplink, which is often the bottleneck due to limited bandwidth compared to the downlink [31], video calls and other RTC applications in the downlink can also experience retransmission-induced non-congestive delays. To address this, an urgency-aware RLC manager that retransmits or skips packets based on slack time can still be applied effectively.

Applicability to emerging RTC applications. Our cross-layer design leverages RTP header information transparently to applications, making PAVE broadly applicable to emerging RTC scenarios. Multi-modal video chat services such as ChatGPT [44] and Gemini [45] use WebRTC APIs, while extended reality platforms from Microsoft [46] are adopting RTP. These applications share similar real-time constraints and can benefit from the optimizations enabled by PAVE.

Non-congestive RAN delays in control planes. PAVE primarily addresses data-plane non-congestive delays in cellular networks. However, control-plane operations, such as mobility management and session establishment, also introduce non-congestive delays [17], [47]. Existing methods for reducing control-plane delays [48], [49] are complementary to PAVE, as PAVE operates solely on data transmission procedures without affecting the control plane of cellular networks.

IX. Related Works

Measurements on cellular networks. Extensive studies have analyzed commercial 5G networks, revealing performance bottlenecks. Prior work investigated cellular uplink performance and bufferbloat issues [50], while recent studies examined bandwidth, latency, and energy consumption in operational 5G networks [31]. However, these primarily focus on link-layer performance and lack a comprehensive analysis of the relations between RAN procedures and RTC applications. Recent cross-layer 5G measurements have identified that specific RAN procedures can degrade RTC performance within time windows as short as 50 ms [17], [51]. In contrast, PAVE captures non-congestive delays introduced by RAN transmission procedures and quantifies their immediate impact on RTC QoE in a frame-level, revealing a dimension overlooked by prior studies.

Improving RTC performance over wireless networks. Extensive research has addressed RTC requirements over cellular networks through various approaches. Deep learning is applied to enhance RTC bitrate adaptation [7], while LRP reduces uplink scheduling procedures from device perspectives [23]. Recent proposals enable wireless infrastructure to directly manage transmission rates [6]. However, these works addressed issues at the device level, which have limitations in reducing non-congestive delays, or only addressed delays caused by congestion. PAVE firstly tackles non-congestive RAN delays for RTC applications on the RAN side.

X. Conclusion

We propose PAVE, a RAN-side mechanism that reduces non-congestive delays for seamless video calls. Our cross-layer measurements on commercial 5G networks revealed that video stalls in RTC applications are often caused by non-congestive delay due to the scheduling procedures and out-of-order packet buffering, rather than network congestion. PAVE leverages RTC applications' inherent periodicity and urgency to enable proactive scheduling and efficient out-of-order packet handling, avoiding the radio resource penalties of existing solutions. Evaluated on Open-RAN testbeds, PAVE achieves higher frame rates with lower video stalls while maintaining resource efficiency. The solution is fully compatible with standard video call protocols, positioning PAVE as a promising Video over NR (VoNR) solution comparable to VoLTE.

REFERENCES

[1] Google, "WebRTC official page," https://webrtc.org/, 2025, [Online; accessed 31-July-2025].

[2] W. DataReportal and Meltwater, "Share of internet users worldwide using their mobile phone to make video calls in the past month as of 1st quarter 2024, by age and gender [graph]," Online, July 2024, available: https://www.statista.com/statistics/1254856/mobile-video-calling-age-gender-distribution/.

[3] Market.us, "Video conferencing statistics 2024," https://scoop.market.us/video-conferencing-statistics/, 2024, accessed: 2025-07-31.

[4] Verizon, "Verizon redefines clear connections on-the-go with the launch of Enhanced Video Calling," https://www.verizon.com/about/news/enhanced-video-calling, 2024, accessed: 2025-07-31.

[5] A. Balasingam, M. Kotaru, and P. Bahl, "{Application-Level} service assurance with 5g {RAN} slicing," in *21st USENIX NSDI 2024*, 2024, pp. 841–857.

[6] Z. Meng, Y. Guo, C. Sun, B. Wang, J. Sherry, H. H. Liu, and M. Xu, "Achieving consistent low latency for wireless real-time communications with the shortest control loop," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 193–206.

[7] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee *et al.*, "Perceive: deep learning-based cellular uplink prediction using real-time scheduling patterns," in *Proceedings of the 18th MobiSys 2020 Conference*, 2020, pp. 377–390.

[8] S. Dhawaskar Sathyanarayana, K. Lee, D. Grunwald, and S. Ha, "Converge: Qoe-driven multipath video conferencing over webrtc," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 637–653.

[9] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 177–192.

[10] M. Poikselkä, H. Holma, J. Hongisto, J. Kallio, and A. Toskala, *Voice over LTE: VoLTE*. John Wiley & Sons, 2012.

[11] I. Lee, S. Kim, S. Sathyanarayana, K. Bin, S. Chong, K. Lee, D. Grunwald, and S. Ha, "R-fec: Rl-based fec adjustment for better qoe in webrtc," in *Proceedings of the 30th ACM MM Conference*, 2022, pp. 2948–2956.

[12] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dapps: Distributed applications for real-time inference and control in o-ran," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 52–58, 2022.

[13] "EdgeRIC: Empowering realtime intelligent optimization and control in NextG networks," in *21st USENIX NSDI 2024*. Santa Clara, CA: USENIX Association, Apr. 2024.

[14] 3GPP TR 38.321, "NR; Medium Access Control (MAC) protocol specification," ver. 18.6.0, 2025.

[15] 3GPP TR 38.322, "NR; Radio Link Control (RLC) protocol specification," ver. 18.2.0, Dec. 2024.

[16] Y. Huang, Y. T. Hou, and W. Lou, "A deep-learning-based link adaptation design for embb/urllc multiplexing in 5g nr," in *IEEE INFOCOM 2021*. IEEE, 2021, pp. 1–10.

[17] F. Yi, H. Wan, K. Jamieson, and O. Michel, "Automated, cross-layer root cause analysis of 5g video-conferencing quality degradation," *arXiv preprint arXiv:2505.14540*, 2025.

[18] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting mobile vr in lte networks: How close are we?" *Proceedings of the ACM SigMetrics 2018*, vol. 2, no. 1, pp. 1–31, 2018.

[19] C. Huitema, "Real time control protocol (rtcp) attribute in session description protocol (sdp)," Tech. Rep., 2003.

[20] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *Proceedings of the ACM MMSys 2016*, 2016, pp. 1–12.

[21] api.video. (2024, Mar.) What is bitrate (and why is bitrate important for your videos). Accessed 31 July 2025. [Online]. Available: https://api.video/blog/video-trends/video-bitrate-importance/

[22] ACCUVER, "XCAL," https://www.accuver.com/sub/products/view.php?idx=6, 2025, [Online; accessed 31-July-2025].

[23] Z. Tan, J. Zhao, Y. Li, Y. Xu, and S. Lu, "Device-based {LTE} latency reduction at the application layer," in *18th USENIX NSDI 2021*, 2021, pp. 471–486.

[24] 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.

[25] srsRAN Project Contributors, "srsran project," https://github.com/srsran/srsran_project, 2025, an open-source 5G software radio suite.

[26] Y. Xie, F. Yi, and K. Jamieson, "Pbe-cc: Congestion control via endpoint-centric, physical-layer bandwidth measurements," in *Proceedings of the ACM SIGCOMM 2020 Conference*, 2020, pp. 451–464.

[27] eBPF Community, "ebpf - extended berkeley packet filter," https://ebpf.io/, accessed: 2025-07-31.

[28] Open5GS Community, "Open5gs: A c-language open source implementation of 5g core and epc, supporting both 5gc and epc," 2024, accessed: 2024-03-12. [Online]. Available: https://open5gs.org/

[29] sysmocom, "sysmocom usim," https://sysmocom.de/products/sim/sysmousim/index.html, 2025, [Online; accessed 31-July-2025].

[30] 3GPP TR 38.211, "NR; Physical channels and modulation," ver. 18.3.0, 2024.

[31] R. A. K. Fezeu, C. Fiandrino, E. Ramadan, J. Carpenter, L. C. De Freitas, F. Bilal, W. Ye, J. Widmer, F. Qian, and Z.-L. Zhang, "Unveiling the 5g mid-band landscape: From network deployment to performance and application qoe," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 358–372.

[32] RC4DAT-6G-60 Programmable Attenuator. https://www.minicircuits.com/WebStore/dashboard.html?model=RC4DAT-6G-60. Accessed: 2025-07-31.

[33] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz, and H. Ahmed, "Understanding and applying precision time protocol," in *2015 Saudi Arabia Smart Grid (SASG)*. IEEE, 2015, pp. 1–7.

[34] Life In Motion Diaries, "Walking man video," YouTube video, 2025. [Online]. Available: https://www.youtube.com/watch?v=C4Z4TWZUqns

[35] Phoebe Beddows, "Driving woman video," YouTube video, 2019. [Online]. Available: https://www.youtube.com/watch?v=m08YfX2YCBQ

[36] I. Lee, J. Lee, K. Lee, D. Grunwald, and S. Ha, "Demystifying commercial video conferencing applications," in *Proceedings of the 29th ACM MM Conference*, 2021, pp. 3583–3591.

[37] Cisco, "Webex - collaboration and customer experience solutions," 2025, accessed: 2025-07-31. [Online]. Available: https://www.webex.com/

[38] GoTo, "Goto meeting web conferencing & online meeting software," 2025, accessed: 2025-07-31. [Online]. Available: https://www.goto.com

[39] Google Meet. [Online]. Available: https://https://meet.google.com/.

[40] 3GPP TR 23.725, "Study on enhancement of URLLC supporting in 5GC," ver. 0.2.0, Jun. 2018.

[41] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*. IEEE, 2017, pp. 1–2.

[42] Google Chrome, "WebRTC Internals - Chrome Debugging Tool," 2025, [Accessed 31-July-2025, only available in Google Chrome]. [Online]. Available: {chrome://webrtc-internals}

[43] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[44] OpenAI. (2025) Connect with webrtc. OpenAI. OpenAI Developer Documentation. [Online]. Available: https://platform.openai.com/docs/guides/realtime#connect-with-webrtc

[45] Google AI. (2025) Get started with live api. Google. Google AI Developer Documentation; last updated 2025-07-08. [Online]. Available: https://ai.google.dev/gemini-api/docs/live

[46] Microsoft. (2024) Holographic remoting version history. Microsoft Learn. Version-history documentation, last updated July 15, 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-version-history

[47] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5g cellular networks," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 86–100.

[48] Y. Li, Z. Yuan, and C. Peng, "A control-plane perspective on reducing data access latency in lte networks," in *Proceedings of the 23rd MobiCom 2017 Conference*, 2017, pp. 56–69.

[49] Y. Li, Q. Li, Z. Zhang, G. Baig, L. Qiu, and S. Lu, "Beyond 5g: Reliable extreme mobility management," in *Proceedings of the ACM SIGCOMM 2020 Conference*, 2020, pp. 344–358.

[50] Y. Guo, F. Qian, Q. A. Chen, Z. M. Mao, and S. Sen, "Understanding on-device bufferbloat for cellular upload," in *Proceedings of the ACM IMC 2016 Conference*, 2016, pp. 303–317.

[51] F. Yi, H. Wan, K. Jamieson, J. Rexford, Y. Xie, and O. Michel, "Athena: Seeing and mitigating wireless impact on video conferencing and beyond," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, 2024, pp. 103–110.