Predictable Real-Time Video Latency Control with Frame-level Collaboration

Jiaxing Zhang[§,¶], Qinghua Wu[§,¶,‡ ✉], Gerui Lv[§,¶], Wenji Du[¶,†], Qingyue Tan[§,¶]
Wanghong Yang[†], Kai Lv[§,¶], Yuankang Zhao[§,¶], Yongmao Ren[†], Zhenyu Li [§,¶,‡], Gaogang Xie [¶,†]
[§]Institute of Computing Technology, Chinese Academy of Sciences, [¶] University of Chinese Academy of Sciences
[†]Computer Network Information Center, Chinese Academy of Sciences, [‡] Purple Mountain Laboratories, China

***Abstract*—Real-time video (RTV) systems place high demands on ultra low-latency (i.e., less than 100 ms). However, our large-scale measurements reveal that a significant portion of users still experience high video frame latency due to bandwidth jitters. Existing solutions attempt to mitigate this issue by lowering the sender's future video frame encoding bitrate. Nevertheless, as shown in our controlled experiments, they fail to drain existing packets queued on the bottleneck node (i.e., the 5G base station and Wi-Fi access point), still suffering from high tail latency as bandwidth decreases.**

In this paper, we propose Co-RTV, a collaborative RTV system that achieves predictable latency control. Specifically, Co-RTV enables endpoint-network collaboration between the bottleneck node and the sender. The collaboration speeds up the release of packets queued at the bottleneck node and facilitates accurate latency control at the RTV sender through scalable QoE-driven flow control. Extensive experiments in emulated networks and on a 5G testbed demonstrate the superior performance of Co-RTV, with tail latency reductions of 69.1% and 70.5%, respectively.

***Index Terms*—real-time video, low latency, collaboration**

I. Introduction

Real-time video (RTV) systems such as cloud-to-vehicle video for auto-driving, cloud gaming, and augmented/virtual reality (AR/VR) [1]–[5] require not only high throughput for high video quality, but also extremely low real-time latency (e.g., less than 100 ms[1] [6]–[8]) to maintain a smooth interactive experience, which is extremely challenging. In RTV systems, the sender runs congestion control algorithms (CCA) [8]–[12] to detect network dynamics and uses flow controllers to determine the encoding bitrate of the video frames, which is based on the bandwidth detected by the congestion controller (shown in Fig. 1). Once the video frames are encoded, they are transmitted through the network channel (e.g., 5G, Wi-Fi, and wired networks) to the receiver. In §II-A and §II-B, we present a detailed latency analysis of existing RTV systems and conclude that *video frame packet queuing at the bottleneck node* (e.g., the 5G base station and Wi-Fi access point) within the network channel is the primary contributor to the RTV system's high latency.

Corresponding author: Qinghua Wu, email: wuqinghua@ict.ac.cn

[1]We apply it from Zhuge [6], Hairpin [7], and Pudica [8]. They observed that user experience degrades significantly once video frame delay exceeds 100ms.

Packet queuing at the bottleneck node (BN) occurs when the RTV sender's encoding bitrate exceeds the network's capacity, for example, due to a bandwidth drop. However, high bandwidth jitter in network channels, especially in wireless networks (e.g., 5G and Wi-Fi), is widely observed due to user mobility and signal blockage [13]–[16]. Consequently, packets are constantly queued up at the BN, typically the 5G base station (BS) and Wi-Fi access point (AP), resulting in increased end-to-end (E2E) latency and reduced quality of experience (QoE) [17], [18]. Since most CCAs in RTV streaming use round-trip time (RTT) as a congestion signal, the inflated queuing latency of the BN delays the congestion signal, greatly challenging the CCA for prompt detection of bandwidth fluctuations. Our large-scale measurements confirm this by observing that a significant portion of users experience high frame delays (§II-A and Tab. I). To achieve RTV's low latency, one solution is to inform the sender of the congestion status at the BN when bandwidth drop, using techniques such as explicit congestion notification (ECN) [19], ABC [20], and L4S [21], [22]. Nevertheless, this type of feedback signal must be processed at the receiver first and thus suffers from a long feedback loop. During this interval, queue buildup at the bottleneck still occurs, meaning that while these solutions improve responsiveness, they remain insufficient for achieving ultra-low latency. Consequently, recent studies focus on shortening the feedback loop by signaling directly from the BN to the RTV sender via the uplink, such as Zhuge [6].

Nevertheless, our controlled experiments (§II-C) reveal that even with the shortest feedback loop, high frame delay is still pervasive in RTV streaming, especially when bandwidth decreases. The root cause is that existing solutions can only *indirectly* drain the packet queue at the BN by reducing the future sending rate (and the frame encoding bitrate). In principle, these methods prevent new packets from being added to the queue, but they *cannot speed up the release of currently queued packets*. As a result, even if the BN knows the exact future bandwidth, the RTV sender still fails to control the latency to meet the stringent low-latency requirements of RTV streaming.

To address this gap, our core idea is to drain the packet queue at the BN as soon as possible when congestion occurs.

The opportunity lies in the fact that RTV systems use the video frame as the basic transmission unit. Therefore, it is feasible to drain the packet queue faster by replacing the currently queued frames (to which these packets belong) with their lower bitrate versions. Specifically, the BN can directly drop frames waiting in the queue, and the sender then re-encodes and retransmits new frames at lower bitrates. While this insight is straightforward, implementing it faces two unique challenges:

(i) How should the BN decide which and how many frames to drop? Because the BN simply forwards each incoming packet to the mobile device, it is not aware of frame-level information such as the frame boundary, i.e., which packets are in the same frame. Furthermore, replacing existing frames with new ones can introduce additional latency, including the frame encoding latency and transmission time from the sender. Therefore, the BN must balance the performance benefits and costs associated with dropping frames.

(ii) How should the sender determine the bitrate of the re-encoded frame? If the sender needs to re-encode the video frames, this indicates that the current latency has become unacceptably high. At this point, the sender CCA's bandwidth estimation is no longer accurate. This greatly challenges the sender in choosing optimal frame bitrates that will quickly clear the packet queue at the base station without significantly degrading video quality.

Clearly, neither simple frame dropping on the network side nor traditional flow control on the RTV sender side can effectively address these challenges alone. We thus envision the collaboration between the RTV sender and the network channel (i.e., BN) to exchange information that each can accurately perceive. The additional information from the sender (resp. BN) will enable the BN (resp. sender) to make the right decision in dropping frames (resp. flow control). Recent developments such as IETF Media over QUIC (MoQ) [23], [24] and IETF Sconepro [25] have demonstrated the feasibility and benefits of the collaboration between endpoints (i.e., RTV sender and RTV receiver) and the network channel. Our solution follows this trend.

In this paper, we propose Co-RTV[2], a systematic collaborative framework for RTV systems. Co-RTV enables **endpoint-network collaboration** between the BN and the RTV sender. The BN relies on the video frame information (e.g., frame boundary and size) from the sender to perform frame dropping. In turn, the sender uses the feedback signal (e.g., bandwidth, queue length, and dropped frames) from the BN to control the bitrate for both newly encoded and re-encoded frames. First, Co-RTV limits the increase in tail latency to a predictable range, ensuring that any rise in latency does not exceed the re-encoding delay cost (§III-B). Second, Co-RTV considers different RTV scenarios' preference to latency and image quality, enabling scalable flow control at the RTV sender (§III-C). In this way, Co-RTV can detect network congestion in advance and precisely control the RTV frame transmission to adapt to bandwidth fluctuations.

[2]"Co" represents "collaborative".

We have implemented and evaluated Co-RTV in emulated networks and on a 5G testbed. Co-RTV significantly reduces tail latency by 69.1% $\sim$ 81.3% in emulated environments and by 70.5% $\sim$ 77% in 5G testbed tests, demonstrating its effectiveness in low-latency RTV streaming.

Our main contributions in this paper are summarised as follows:

- We uncover the performance issue of RTV systems in large-scale production networks and the limitations of existing studies, and reveal the root cause (packet queuing at the BN increases the latency). (§II).
- We design Co-RTV as a collaborative framework between the BN and the sender, performing predictable latency control and scalable flow control (§III).
- We have implemented and extensively evaluated Co-RTV in emulated networks and on a 5G testbed, demonstrating its superior performance (§IV and §V).

II. Background and Motivation

A. Background: RTV system's high latency

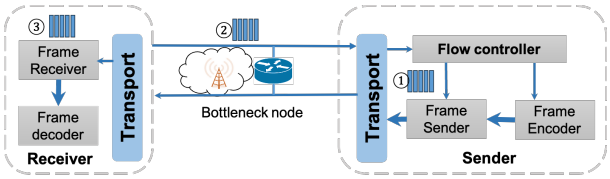


Fig. 1: Queues in RTV system

RTV systems demand low-latency ($<$ 100 ms) video streaming to deliver an immersive user experience. In the RTV system, the receiver decodes frames and renders video images, and the sender is usually deployed on a cloud server and runs a flow/congestion controller to dynamically adapt the encoding bitrate and sending rate to the changing network bandwidth [8]–[11], [26]. However, due to bandwidth fluctuations and limitations of the encoder and decoder, queuing occurs at various points: the sender (queue ①), BN (queue ②, e.g., the 5G BS and Wi-Fi AP), and the receiver (queue ③), which significantly increase latency, as shown in Fig. 1.

Queue ① and Queue ③ have already been well controlled. Queue ① arises from a mismatch between the sending rate and the frame bitrate, typically due to encoder overshoot [27], [28] or encoding delays (during the encoding period, the sending rate is reduced). Fortunately, this issue can be mitigated using advanced hardware encoders [29] or by employing Salsify [27], which can quickly drain queue ① through an efficient encoder that can store encoding states. Queue ③ at the receiver occurs when the encoding rate exceeds the receiver's decoding capacity, a problem that can be addressed by techniques such as AFR [30], [31], which adjusts the frame rate to match decoding capability.

Queue ②, however, presents a significant challenge in current RTV systems. It usually occurs due to the sending rate exceeding the BN's forwarding rate or the sudden arrival of other users' burst flows. Fortunately, emerging networks

represented by 5G, Wi-Fi 6, and Wi-Fi 7 offer multi-user assurance [32]–[42] (detailed in §VI), which means other users' burst flows will not increase the length of queue ②. Therefore, there is a possibility to solve it by a well-designed flow controller that can perceive or predict bandwidth fluctuations in a timely manner.

Nevertheless, designing an RTV flow controller that works well is challenging. This is mainly due to frequent bandwidth fluctuations. In 5G and Wi-Fi networks, mobility and channel interference often cause bandwidth fluctuations [14]–[16], [43]–[45]. In extreme cases, bandwidth can drop by up to 10x at the 99th percentile [6]. To detect the bandwidth and avoid the formation of queue ②, the sender relies on the E2E feedback signals, such as acknowledgment (ACK), negative acknowledgment (NACK), or RTCP feedback [46]. However, once queue ② forms when the bandwidth drops, it obstructs the arrival of subsequent packets at the RTV receiver, delaying the generation and return of feedback signals. As a result, the RTV sender cannot react promptly to bandwidth reductions, leading to extended queue ② and increased E2E latency, a phenomenon referred to as the *inflated control loop* [6].

We summarize the above analysis with an examination of over 10 million flows from a real-world RTV system (anonymous for legal reasons) in Table I, which confirms the analysis that RTV in wireless networks (especially 5G and Wi-Fi 2.4 GHz) suffers from high frame delay and a significant portion of users are impacted by the high latency.

	Wired	Wi-Fi (5 GHz)	Wi-Fi (2.4 GHz)	5G
Frame	1.28	3.42	4.9%	2.1%
User	2.9%	5.7%	29.1%	15.6%

TABLE I: Proportion of frames experiencing delay $>$ 100 ms and proportion of users with over 5% of frames delayed over 100 ms across different networks in a real-world RTV platform.

B. Investigation of RTV system's high latency

To uncover the root causes of the high latency of RTV systems, we further investigate the inherent limitation of the E2E congestion control solution and explain the impact of queue ② on RTV latency. We conducted a week-long continuous RTV playback for A/B tests in our lab. Using a 5G network as an example, both a wired network and a B210 5G base station were connected to the same network, with the sender employing COPA [9], a delay-based congestion control algorithm. Moreover, to mitigate the influence of queue ① and queue ③, we incorporated the Salsify [27] into our encoder and utilized a hardware decoder in the receiver in subsequent experiments.

The experimental results indicate that the proportion of frames with high latency (>100 ms) was 0.13% in the wired network and 1.37% in 5G. We recorded both the per-frame total delay and the queued delay at the BS, and report in Fig. 2. The results reveal the correlation between the high tail frame delay and the base station's queued delay, indicating that the bottleneck primarily residing in the wireless segment between

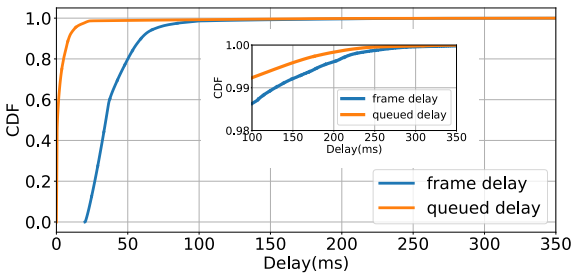


Fig. 2: Total frame delay and the queued delay. High tail frame delay is caused by queuing at the BS.

the user and the 5G BS. We further zoomed in on a high-latency test case of one second during the RTV transmission, which presents frame delay and queued delay, as well as the sender's encoding rate and sending rate, and the BS's forwarding rate and queued delay, as shown in Fig. 3. It can be seen that delayed congestion control struggles to drain the queue ②, resulting in increased station's queued delay.

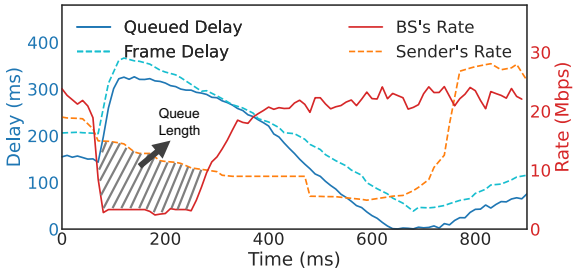


Fig. 3: A case of the sender and the BS's states. The sender takes over 300 ms to decrease to a reasonable sending rate when the bandwidth drops. The RTT between the BS and the sender is about 40 ms.

C. Limitations of existing solutions

To mitigate the impact of Queue ② on RTV systems, several efforts have been made to make the flow controller closer to an oracle-designed flow controller, such as shortening the control loop [6], [19], [20], [47] to reduce congestion perception time. The recently emerging L4S [48]–[50] optimizes latency with ECN while also controlling queued time. Nevertheless, we observe that existing solutions still suffer from high latency in RTV streaming, especially when bandwidth drops, as illustrated in Fig. 4. In this controlled experiment, we use Linux TC to emulate 5G and Wi-Fi bandwidth fluctuations (E2E RTT is set to constant 60 ms; other details are in §V) and choose five solutions for comparison: *(i)* COPA [9], a delay-based congestion control algorithm; *(ii)* ABC [20], where the sender uses a bandwidth "increase" or "decrease" signal marked by the BN and processed by the receiver to determine the sending rate; *(iii)* Zhuge [6], where the sender detects congestion by delayed ACKs directly from the BN via the uplink; *(iv)* L4S [48]–[50], where the sender is fed with the congested packet rate from the receiver and keeps

the BN's packet queue length small via dropping packets[3]; *(v)* Feedback-Driven Congestion Control (FDCC), where the RTV sender is fed with the *actual future bandwidth in traces* from the BN. Note that we introduce FDCC as an oracle flow control solution (detailed in §V-A), it operates the *shortest control loop*. However, since the feedback takes at least one RTT delay (BN to sender) to take effect, this can also lead to queue buildup.

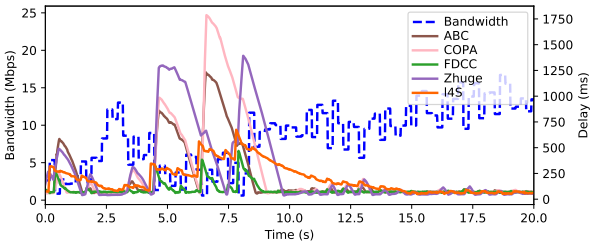


Fig. 4: Frame delay of different solutions in emulated networks.

Fig. 4 shows that under the same bandwidth trace, different solutions experience large differences in frame delay. The average frame delays of COPA, ABC, Zhuge, and L4S are 213.3 ms, 173.5 ms, 227 ms, and 163 ms, respectively. These averages are well above the stringent 100 ms low latency requirements of RTV applications [7], [8]. We also found that these solutions experience severe delay spikes (up to 1787 ms, 1231 ms, 1392.5 ms, and 674.5 ms, respectively) when bandwidth drops. In contrast, FDCC significantly reduces the average delay to 87.1 ms.

Nevertheless, it is notable that FDCC still has 13.1% of frames with an E2E delay exceeding 100 ms. These results indicate that: even the solutions with the shortest control loop struggle to meet the low latency requirements of RTV streaming. Although L4S keeps the length of the queue ② small by dropping incoming packets, the sender will retransmit the packets dropped by the BN to maintain normal frame decoding and playback at the receiver [4], which makes it difficult to achieve low latency. Assuming that the time between the bandwidth drop and its effect at the sender is T and that the available bandwidth drops to $\frac{1}{n}$. By the time a new video frame (affected by the feedback) reaches the BN, the accumulated queue takes $2(n-1)T$ to drain. This also explains the reason of high tail latency when the bandwidth drops by more than 10×.

In summary, existing solutions that control latency by reducing the future encoding bitrate cannot immediately release the current frames queued at the BN. Even if the sender can adjust its output bitrate at the exact moment of congestion, it may still take a long time to drain the existing queue at the BN, ultimately resulting in unsatisfied E2E latency.

[3]On one hand, the feedback signal in L4S needs to pass through the receiver, resulting in a longer feedback path that relies on ECN signals. On the other hand, L4S promises low packet loss, but its queue is relatively shorter. If feedback is delayed and the queue fills up, this also results in packet loss.

[4]Video frames are compressed using reference frames during encoding. If reference frames have not arrived, subsequent frames cannot be decoded.

D. Opportunity: endpoint and network collaboration

An intuitive solution to reduce the queued delay is to speed up the release of queued packets at the BN (queue ②) when congestion occurs. This idea can be accomplished by directly replacing queued frames with their lower bitrate versions. Specifically, the BN can drop queued frames, and the RTV sender then re-encodes and retransmits these frames at lower bitrates (as in Salsify [27]). Nevertheless, this simple idea faces two practical challenges:

- *Balancing latency and quality*: while re-encoding low-bitrate frames promises to reduce latency, it may sacrifice video quality, which also affects user QoE. This poses two requirements. Firstly, the BN must carefully decide whether to drop frames and how many frames to drop. Secondly, the sender must carefully choose the bitrate for the re-encoded frames or skip some of them according to the QoE requirements of various RTV applications.
- *Accommodating the loss recovery logic*. The sender transport layer performs reliable transmission mechanisms such as loss detection and retransmission. If the sender has no knowledge of the frames dropped by the BN, it will continuously retransmit packets in those frames, wasting bandwidth and traffic.

To overcome the above challenges, it is necessary to coordinate the BN with the RTV sender. On the one hand, the BN needs frame boundary information (i.e., which packet belongs to which frame) from the sender to drop frames. On the other hand, the sender needs feedback from the BN about the dropped frames and network dynamics to re-encode or skip frames. With frame dropping enabled at the BN, we can limit the uncontrollable tail latency increase to less than the time required to replace queued frames, which is equivalent to the feedback and re-encoding time. Moreover, the direct feedback from the BN paves the way for precise flow control at the sender, addressing the major drawback of existing solutions (i.e., unable to perceive in-flight frame status timely). Note that recent developments such as IETF media over quic (MoQ) [23], [24] have demonstrated the feasibility and benefits of the collaboration between endpoints and the network channel. The above insight leads to our design of Co-RTV, a collaborative flow control framework that achieves predictable tail latency control for RTV streaming.

III. Co-RTV Design

This section details the design of Co-RTV. Overall, Co-RTV is composed of two core components: 1) predictable latency control via collaborative frame-drop at the BN and 2) scalable QoE-driven flow control at the RTV sender. We begin with an overview of Co-RTV in §III-A, followed by the details of both components in §III-B and §III-C.

A. Co-RTV overview

Co-RTV is designed to meet two primary goals: predictable latency control and scalable QoE-driven flow control.

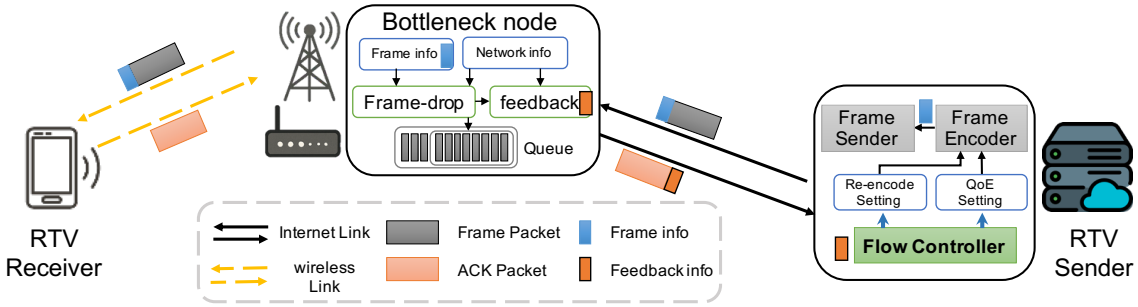


Fig. 5: The overview of Co-RTV

Predictable tail latency control: By implementing the collaborative control, Co-RTV confines the increase in tail latency within a predictable range, ensuring that the latency increase does not exceed the re-encoding delay cost (described as $T_i' + Q_{di}$ in §III-B1 and further reduced by T_{pre} in §III-B4). This precise control over tail latency distinguishes Co-RTV from prior approaches.

Scalable QoE-driven flow control: Co-RTV employs a scalable flow control to manage predictable latency increase, customized to meet the QoE requirements of various RTV scenarios. Specifically, Co-RTV considers the scenario preferences to latency, frame rate, and image quality, enabling flexible QoE configurations to optimize various RTV scenarios' flow control strategies.

Fig. 5 illustrates the architecture of Co-RTV. In Co-RTV, the BN has access to the frame information (e.g., frame size and boundaries) embedded by the sender, which is detailed in §III-B2. Upon receiving video frames, the BN estimates the frame forwarding delay and performs frame-drop control for frames with excessive forwarding delays to make the latency inflation predictable. Subsequently, the BN provides feedback to the RTV sender, which adjusts its flow control strategy by modifying (re-)encoding settings based on the QoE requirements of the scenarios to make the flow control scalable. A summary of the key parameters used by Co-RTV in §III-B and §III-C is provided in Tab. II.

Parameter	Explanation
α	sensitivity parameter representing the quality decrease
w	The time window for calculating the bandwidth at the BN
$Gain_i$	the gain of dropping frame i
L_{di}	latency decrease of dropping frame i
Q_{di}	quality decrease of dropping frame i
Bw	the BN's bandwidth
RTT_{BN}	the RTT between the BN and the sender
$QLen$	the queue length of the BN
FR	frame rate
Bw_{old}	the BN's bandwidth before the bandwidth decrease
T_{pre}	time of decision-making prior to the frame arrival
$size(i)$	the size of frame i
$SndRate$	the sending rate of the sender
R	the encoding bitrate of the sender
τ	the queue empty time setting of the sender
QSF	the quality sensitivity factor setting of the sender
N_s	number of skipped frames
N_d	number of dropped frames

TABLE II: Explanation of parameters

B. BN: predictable tail latency control

At the BN, frame-drop control plays a key role in predictable tail latency control. The BN continuously estimates the forwarding delay for each video frame using the frame information and network information. It then decides whether to drop the frame based on its potential impact on latency and video quality reduction. In §III-B1, we introduce the decision principle of frame-drop and the required information. §III-B2 details the required frame information embedded by the sender. Later, §III-B3 and §III-B4 present Co-RTV's bandwidth smoothing strategy for robustness and the proactive frame-drop control to ensure real-time performance; these play a key role in Co-RTV for timely and effective frame-drop under bandwidth fluctuations. Last, §III-B5 introduces the feedback sent to the RTV sender.

1) Principle of frame-drop decision: For a frame i, a frame-drop operation results in two impacts:

- Latency decrease (L_{di}): Dropping a large frame reduces the forwarding time, which can be significant in cases where the frame is too large.
- Quality decrease (Q_{di}): Dropping a frame leads to a reduction in quality, as the new frame is usually re-encoded to a lower bitrate or skipped.

L_{di} is quantified in seconds, while Q_{di} is evaluated using an equivalent time increase metric, also quantified in seconds (see Eq. 4). Therefore, the gain of dropping frame i, denoted $Gain_i$, can be defined as:

$$Gain_i = L_{di} - Q_{di} \tag{1}$$

The BN can compute the above gain $Gain_i$ for each video frame, then drop the frames with $Gain_i > 0$. For a frame i, L_{di} can be computed by the frame i's forwarding time T_i and the re-encoded frame's forwarding time T_i'.

$$L_{di} = T_i - T_i' \tag{2}$$

The BN tracks the current queue length $QLen$ and forwarding bandwidth Bw. T_i can be easily calculated if the size of each frame $size(i)$ is known. As shown in Fig. 6, calculating T_i' is more complex. Initially, re-encoding feedback information must be sent back to the sender, which takes 1/2 RTT_{BN}. The sender then needs some time for re-encoding (no

more than $\frac{1}{FR}$), after which the frame is transmitted back to the BN, requiring another 1/2 RTT_{BN}. Due to pacing during transmission, the complete frame will take no more than $\frac{1}{FR}$ time to be sent over. Here, RTT_{BN} represents the round-trip time between the BN and the sender, which is detailed by RTT_{BN} calculation in supplemental materials, while FR denotes the video frame rate. L_{di} then can be represented as:

$$L_{di} = \frac{QLen + size(i)}{Bw} - (RTT_{BN} + \frac{2}{FR}) \tag{3}$$

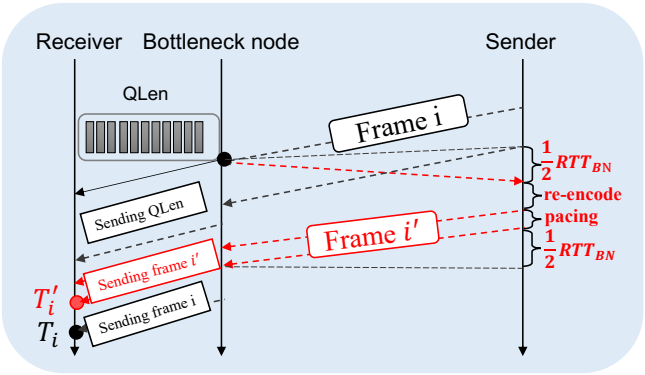


Fig. 6: Illustration of forwarding i or re-encoding i'

The re-encoding quality decrease Q_{di} of frame i is related to the bandwidth drop ratio. Let Bw_{old} represent the bandwidth before the reduction and Bw represent the current bandwidth. Q_{di} can then be expressed as:

$$Q_{di} = \alpha \cdot \frac{Bw_{old} - Bw}{Bw} \cdot \frac{1}{FR} \tag{4}$$

where $\alpha < 1$ is a design parameter to configure the sensitivity of frame dropping. We set α to 0.5 in our current implementation based on numerous experiments.

Consequently, the cost of dropping a frame i is $T_i' + Q_{di}$, which can be interpreted as the threshold for tail latency growth. In summary, the BN can make real-time frame-drop decisions by analysing the frame information, BN queue information, and bandwidth.

2) Frame information embedded at the RTV sender: To enable the cost computation of frame dropping, each packet sent by the sender is embedded with the following frame information.

- Frame Number: Identifying the sequence of the video frame.
- Packet Sequence: Specifying the packet's position within a specific frame.
- Total Packets Number: Indicating the total number of packets of a specific frame.
- Latest Re-encoded Frame Number: Tracking the most recently re-encoded frame, which is used for forwarding recovery in §III-B4.

One way to implement the piggyback of the above information is using media over QUIC (MoQ) [23], [24] as the latest IETF MoQ draft supports carrying relevant media information.

3) Bandwidth estimation at the BN: The BN is aware of each user's amount of data successfully transmitted over a period of time, enabling Co-RTV to accurately calculate the available bandwidth Bw based on previous efforts [20], [33], [47], [51]–[57]. However, the unsmoothed Bw fluctuates severely, which significantly disrupts the real-time decision-making of Co-RTV. The root cause lies in the unpredictability of channel quality and other users' flow patterns, which affect network capacity allocation. Previous solutions [6], [20] typically rely on a longer time for smoothing. Smoothing effectively reduces bandwidth fluctuations but also delays congestion awareness. This trade-off may fail to meet RTV's real-time requirements, a point not discussed in previous work. Indeed, simultaneously meeting real-time requirements and bandwidth calculation robustness is a significant challenge. Co-RTV employs the average forwarding bandwidth over a time window w for smoothing to enhance the robustness of bandwidth calculation. However, this smoothing technique may introduce some inaccuracies in bandwidth estimation. To address this issue, Co-RTV continuously provides $QLen$ feedback to the sender to mitigate the impact of bandwidth overestimation (Eq. 6). Our experiments in §V-B demonstrates that the latency can still be optimized even with some deviation in bandwidth. Furthermore, Co-RTV integrates proactive frame-drop control (detailed later) to improve the responsiveness. Our current implementation sets w to 50 ms based on real-world deployments.

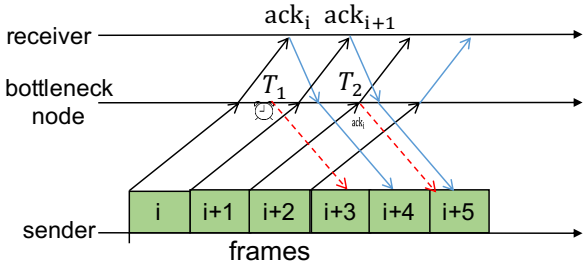


Fig. 7: Case study of dropping frame $i+2$

4) Frame dropping at the BN: With the frame and network information, calculating the $Gain_i$ is straightforward. Then the question nails down to the time for the calculation of the cost related to frame dropping and the operation of frame dropping. A straightforward approach is to calculate when the frame reaches the BN. However, this approach delays frame-drop decision-making if the dropped frames arrive later than the bandwidth reduction. Therefore, Co-RTV makes frame-dropping decisions k frames ahead of the frame arrival. An example with $k = 2$, illustrating the dropping of frame $i+2$, is shown in Fig. 7.

Let us consider a scenario at time T_1 when a bandwidth drop occurs while transmitting frame i. This reduction is not severe enough to cause significant queue buildup. Hence, the frames i and $i+1$ are unlikely to be dropped, while frame $i+2$ will be dropped. Instead of evaluating whether to drop frame $i+2$ when it arrives at the BN at time T_2, Co-RTV proposes a

more proactive approach: *predict and decide which frames to drop immediately when the bandwidth drops at time* T_1. This approach is motivated by the fact that the earlier the decision to drop the frame is made, the sooner the RTV sender receives the feedback from the BN for proper flow control.

To achieve this, the BN must estimate the frame size and queue length of frame $i+2$ before it arrives. In most RTV scenarios, such as cloud gaming and video conferencing, I-frames are transmitted only for synchronization during the initial transmission or after a prolonged network disconnection, while the rest of the video frames are P-frames. Hence, the size difference between frames i, $i+1$, and $i+2$ is minimal (all are P-frames). As such, the size of frame $i+2$ can be inferred from frame i. In conclusion, using the bandwidth at T_1 and the size of frame i, we can predict the queue length at $T2$ and make the frame-drop control in advance. With this basis, L_{di} can be represented as follows, where T_{pre} represents the time of decision-making made prior to the frame arrival time.

$$L_{di} = \frac{QLen + size(i)}{Bw} - (RTT_{BN} + \frac{2}{FR}) + T_{pre} \quad (5)$$

We can then calculate the $Gain_i$ for each frame i and perform the frame-drop control algorithm as depicted in Algorithm 1.

Algorithm 1: Frame-drop Algorithm

Data: Network Information: RTT_{BN}, $QLen$, Bw, Bw_{old}; Frame information: FR, $size(i)$, T_{pre}
Result: BN's frame-drop list: L

```
function OnFrameDropping():
    L ← {};
    for frame i in framelist and Bw_old > Bw do
        L_di ← (QLen+size(i))/Bw − (RTT_BN + 2/FR) + T_pre;
        Q_di ← α · (Bw_old − Bw)/Bw · 1/FR;
        Gain_i ← L_di − Q_di;
        if Gain_i > 0 then
            L ← L ∪ {i};
        end
    end
    return L
```

Forwarding recovery after frame dropping: Once deciding to drop frame i, the BN continuously monitors the "Frame Number" field and the "Latest Re-encoded Frame Number" field as detailed in §III-B2. If the "Frame Number" $< i$ or the "Latest Re-encoded Frame Number" $>= i$, the video frame can be forwarded. Here, "Frame Number" $< i$ indicates that the frame is one of the video frames before frame i that should be dropped, while "Latest Re-encoded Frame Number" $>= i$ means the frame is a newly encoded video frame after the sender receives the dropping feedback of frame i. In other words, newly encoded video frames should be forwarded, enabling the timely recovery of the forwarding state.

5) Feedback to sender: After determining which frames to drop, the BN sends feedback to the sender. The feedback includes:

- Bandwidth: The current bandwidth of the BN.
- Queue Length: The current queue length of the BN.
- Dropped Frame Information: The sequence number of the frame that starts to be dropped.

If no frames are dropped, the feedback simply contains the queue length. In cases where frames are dropped, the feedback specifies which frames were dropped and notifies the sender to adjust its encoding strategy for subsequent frames. The feedback is transmitted at intervals aligned with the RTT between the BN and the sender. As shown in TECC [58], sending feedback once per RTT_{BN} is sufficient. Note that if frame-drop occurs, the feedback will be sent immediately.

C. Sender: scalable QoE-driven flow control

The second core component of Co-RTV is the scalable QoE-driven flow control at the RTV sender. Upon receiving the feedback of frame dropping, the sender must decide the bitrate and whether to re-encode or skip some of the dropped frames based on the QoE requirements. In this section, we present Co-RTV's scalable flow control to optimize the QoE for different RTV scenarios.

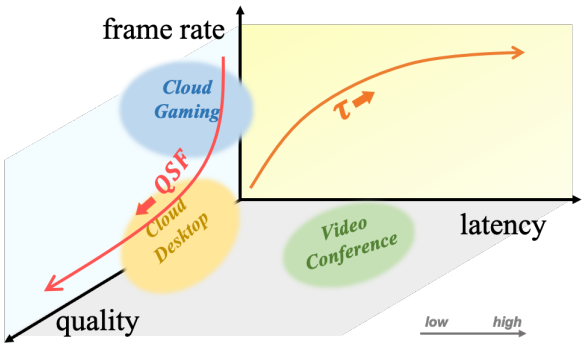


Fig. 8: Different scenarios' preference for latency, frame rate, and quality.

1) Scalable flow control design space.: As shown in Fig. 8, different RTV scenarios have different priorities in reducing bandwidth consumption when bandwidth drops or queues accumulate. To accommodate these varying QoE requirements, Co-RTV introduces two scalable parameters: *queue empty time* and *quality sensitivity factor*, where the queue empty time represents the trade-off between latency and bitrate (quality $\times$ frame rate), while the quality sensitivity factor captures the trade-off between frame rate and quality. The effects of the two parameters are also plotted in Fig. 8.

Queue Empty Time (τ): When the BN's queue begins to accumulate, the sender needs to reduce its bitrate to drain the queue. The time to empty the queue directly impacts the latency of video frames. We denote the queue empty time as τ. As shown in Eq. 6 and Fig. 8, a longer empty time results in a slower decrease in bitrate, affecting a larger number of video frames and increasing their latency. Conversely, a shorter

empty time accelerates the bitrate reduction, impacting fewer frames, while it leads to the generation of some extremely small frames.

Quality Sensitivity Factor (QSF): Bandwidth drop and frame dropping operations by the BN result in a bitrate decrease, while the trade-off between the frame rate and the quality varies across scenarios [59]. QSF represents the balance between frame rate and quality. As shown in Fig. 8, a lower QSF favors maintaining the frame rate, while a higher QSF prioritizes maintaining quality. This flexibility allows the RTV sender to adapt the encoding strategy according to its specific QoE preferences.

2) Encoding strategy: The encoding strategy at the sender is determined with the feedback from the BN and its own QoE requirements, particularly focusing on how the queue empty time (τ) and the quality sensitivity factor (QSF) influence the encoding behavior.

Congestion control: Co-RTV employs a traditional delay-based CCA. In our implementation, we use COPA, which can be replaced with other CCAs. We further introduce an optimization to address the impact of non-congestion delay jitter in wireless networks, thereby enhancing Co-RTV's throughput without unnecessary bitrate reductions. Specifically, when the sender observes an increase in RTT within a threshold while the BN's queue length remains nearly unchanged, it maintains the current bandwidth instead of reducing it. In contrast, after receiving the feedback of frame dropping, the bandwidth will be updated to that fed back by the BN (Bw in §III-B5).

Encoding Bitrate Calculation: When receiving feedback, the encoding bitrate (R) is simply calculated by the queue length ($QLen$), the sending rate ($SndRate$) from the congestion controller, and the queue empty time (τ). It ensures the sender reduces its bitrate to drain the BN queue without overburdening the network, which is known as queue-avoidance.

$$R = SndRate - \frac{QLen}{\tau} \tag{6}$$

Upon receiving frame-drop feedback, the number of dropped frames, denoted as N_d, is determined by the sequence number of the dropped frame, i, and the sequence number of the next frame to be encoded, j, such that $N_d = j - i$. Since the dropped video frames need to be re-encoded and re-sent, they will occupy the bandwidth that would otherwise be used for normally encoded video frames. The encoding bitrate, R, is subsequently adjusted as:

$$R = R \cdot \frac{\tau \cdot FR}{N_d + \tau \cdot FR} \tag{7}$$

QoE-driven Frame-skipping Strategy: As lowering the bitrate alone sacrifices the frame quality, we use the Quality Sensitivity Factor (QSF) to make a trade-off between frame quality and frame rate. The number of frames to be skipped, N_s, is calculated with QSF:

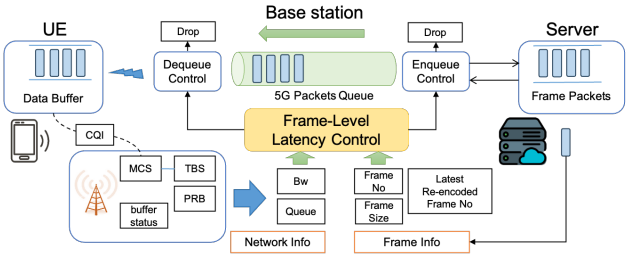


Fig. 9: Co-RTV implementation in 5G

$$N_s = QSF \cdot (\tau \cdot FR + N_d)\left(1 - \frac{R}{SndRate}\right), \quad 0 \le QSF \le 1 \tag{8}$$

Here, $QSF = 1$ indicates the scenario prioritizes maintaining high video quality, while $QSF = 0$ indicates that the scenario favors maintaining the frame rate without skipping frames. Once the number of skipped frames is determined, the skipped frames are evenly distributed across the future encoded frame sequence to ensure smooth playback. After skipping frames, the sender will increase the bitrate R to ultilize the preserved bandwidth of skipped frames.

IV. Co-RTV Implementation

Although implementing Co-RTV on Wi-Fi access points is relatively straightforward, we first deploy Co-RTV in a 5G environment. This choice is motivated by scenarios such as industrial Internet and autonomous driving, where wired connections are impractical and Wi-Fi coverage is insufficient. Below, we detail implementation of Co-RTV at the 5G base station and the feedback implementation, followed by the RTT_{BN} calculation and Co-RTV's encoder.

At last, for more implementation about RTT_{BS} calculation and Co-RTV's encoder, please refer to supplemental materials.

Practical implementation: As illustrated in Fig. 9, we establish an E2E 5G network testbed in our laboratory. Both the 5G core network and the radio access network (RAN) stacks are deployed using the open-source OpenAirInterface (OAI) [60]–[62] 5G project, which is compliant with the 3GPP Release 15 standard [35]. The Co-RTV server and 5G core components are hosted on a cloud server. The 5G base station is built on an SDR-based (Software-Defined Radio) device, XG-Station [63], which supports customized physical layer configurations. Leveraging the 5G RAN design, the base station determines the capacity allocated to each mobile user by accessing physical layer resource allocation. Following the approaches of PBE-CC [47] and Tutti [53], we select two key resource parameters, Physical Resource Block (PRB) and Transport Block Size (TBS), to estimate both capacity and bandwidth. Based on these metrics, Co-RTV performs frame-drop control through a two-phase mechanism: enqueue and dequeue control. Further details regarding code-level deployment are provided in the supplementary materials.

Implementation of feedback and frame information embedding: Information exchange between the network and the endpoint is deeply discussed in IETF MoQ [23], [24],

IETF Sconepro [25] and XCP [64], which becomes the trend in next-generation networks. As more secure and effective feedback channels/methods will appear, we will update Co-RTV accordingly. To verify the effectiveness as well as enable collaboration between the sender and the BN, Co-RTV extends the APN6 [65] framework for enhanced feedback and frame information in IPv6. For IPv4, Co-RTV supports both the RTP/RTCP and QUIC, with the feedback information placed at the beginning of the UDP datagram. It is important to note that Co-RTV currently does not support the TCP protocol, as the streaming transmission mechanism of TCP does not allow for reverting to an untransmitted state once data has been sent. Moreover, most RTV systems don't prefer to use TCP due to its head-of-line blocking problems [66]. However, modifying the server-side TCP protocol to support Co-RTV in the future remains a possibility.

RTT_{BN} **calculation:** The BN can estimate the RTT from the BN to the user endpoint based on link-layer data. Additionally, by obtaining the RTT between the client and the sender from the RTCP packets [6], the BN can calculate RTT_{BN}. For the QUIC protocol, RTT_{BN} can be estimated by real-time monitoring of the packet number of uplink packets and ACK frame's contained packet number of downlink packets. To achieve this in the actual deployment, we removed the encryption of the QUIC packet number and the ACK frame, which poses minimal security risk as TCP's packet number and the ACK frame are in plaintext. If maintaining E2E encryption is desired, the E2E RTT information can be conveyed through the IP options field or the MoQ standard protocol to calculate RTT_{BN}, where $RTT_{BN} = RTT_{E2E} - RTT_{LL}$ (with RTT_{LL} representing the link-layer RTT between the BN and the RTV receiver).

Co-RTV's Encoder: At the sender side, Co-RTV implements advanced encoding state preservation and recovery mechanisms, similar to Salsify [27]. The encoder saves and restores its internal state based on the hardware encoders. The encoder adjusts its reference frame and bitrate with the saved encoding state to ensure that the stream remains consistent without reinitializing the entire encoding process when receiving the feedback of frame dropping.

V. Evaluation

In this section, we present the evaluation of Co-RTV, which consists of two parts: emulated environment evaluation and 5G testbed evaluation.

Emulated environment evaluation: We deployed a Linux TC-based emulation to test Co-RTV's performance. The testbed comprises three main docker containers: an RTV receiver container using the XQUIC [67] protocol which is a cross-platform implementation of QUIC [68], [69], an emulated BN container to emulate 5G network fluctuations via 5G traces as well as control packet forwarding, and an RTV sender container running the RTV and XQUIC stack.

5G testbed evaluation: The 5G testbed experiments were conducted using our 5G implementation and a deployed Co-RTV sender on the cloud. The measured RTT from the 5G base station to the cloud Co-RTV sender was approximately **60 ms**. The Co-RTV client connected to a 5G Customer Premises Equipment (CPE) device and accessed the sender via the 5G base station, while other users accessed the Internet through separate CPEs. Random network activities from these users introduced random traffic variability, providing a realistic environment to evaluate Co-RTV's performance.

A. Experiment Setup

We evaluated Co-RTV by transferring the video at 60 fps. In subsequent experiments, forward error correction (FEC) was not enabled in the sender. We introduce the baselines, traces, and evaluation metrics used in the experiments below.

Baselines. The baseline algorithms include COPA, ABC, Zhuge, FDCC, and Co-RTV without feedback.

- COPA [9]: A widely used delay-based RTV congestion control algorithm integrated with XQUIC [67].
- ABC [20]: A feedback-driven solution where the BN predicts bandwidth variations and informs the sender bandwidth "increase" or "decrease" signal.
- Zhuge [6]: A feedback solution where the BN informs the sender of congestion via delayed ACKs.
- FDCC: Operating as an oracle congestion control in emulated networks and on our 5G testbed by feeding precise bandwidth Bw and queue length $QLen$ to the sender. The sender calculates its sending rate $Rate$ based on this information as:

$$Rate = Bw - \frac{QLen}{\tau} \tag{9}$$

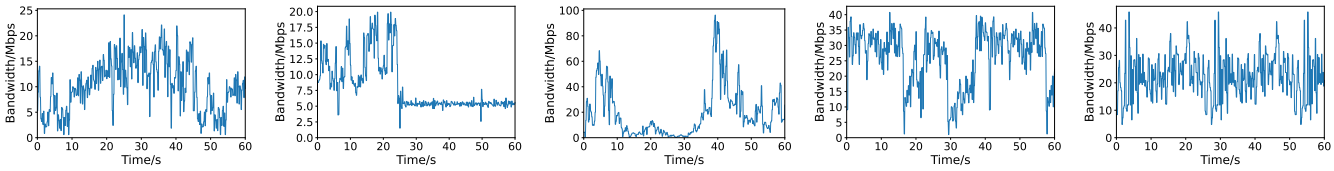
  where τ represents empty time, the same role as in Co-RTV.
- Co-RTV without feedback (w/o fb): the BN drops frames when bandwidth drops without feedback. The sender doesn't retransmit them by an advanced reference frame setting referred to Salsify [27].

Note that we did not include additional comparisons with L4S [21], [22] and other CCAs such as SQP [10], GCC [11], BurstRTC [26], and Pudica [8]. This is because we believe that FDCC represents the theoretical upper bound achievable by these CCAs and L4S. Furthermore, since L4S operates based on SCREAM [70], comparisons with COPA and COPA-based solutions (such as ABC and Zhuge) would not be meaningful.

Traces. The evaluation utilized five weak long network traces collected in real-world 5G environments from shopping malls, classrooms, outdoor areas, and stadiums with 5G Android phones using iperf3. These traces are detailed in supplemental materials, denoted as $mall$, $office$, $classroom$, $outdoor$, and $stadium$, as shown in Fig. 10.

Metrics. We use the following metrics for evaluation.

- Frame delay. The time needed between frame encoding and decoding, reflecting real-time latency.
- Frame size. An indicator of video image quality, with larger frames indicating higher visual fidelity.



(a) Mall (b) Office (c) Classroom (d) Outdoor (e) Stadium

Fig. 10: Network traces

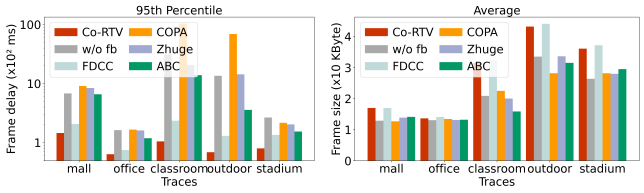


Fig. 11: Overall performance under different traces

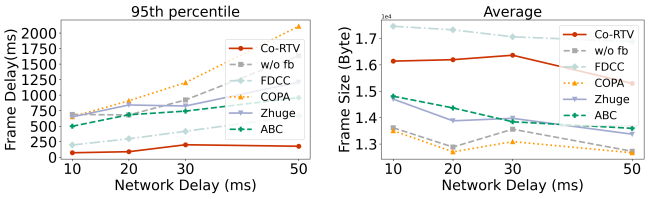


Fig. 12: Performance under different network delays

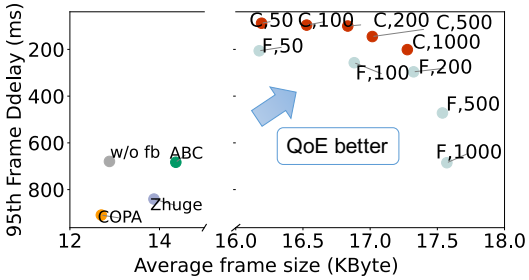


Fig. 13: Different empty time

B. End-to-End Performance

We first evaluate the performance of Co-RTV in emulated environments. The RTT from the receiver container to the BN container is set to *fixed 10ms*[5] in later experiments since the RTT between the real-world user endpoint and the BN is about 10 ms. The RTT from the BN container to the sender container changes in later experiments. Unless otherwise specified, the empty time τ in both FDCC and Co-RTV was set to 50ms, the QSF was set to 0, and the delay between the BN and the sender is set to 20 ms under trace $mall$.

Overall performance. *As shown in Fig. 11, Co-RTV reduces 95th percentile frame delay by over 69% and increases average video frame size by 28.2% $\sim$ 36.3% than other solutions.* Compared to FDCC, Co-RTV reduces the frame delay by 37.5% while achieving similar frame quality. As shown in the 95th percentile delay results, Co-RTV maintains a tail latency of around 100 ms across diverse networks, offering significantly lower latency than FDCC as well as other solutions. In terms of frame quality, the comparison of frame sizes shows that Co-RTV and FDCC both achieve high video quality, with little difference between them (within 2.1%), yet Co-RTV outperforms other solutions.

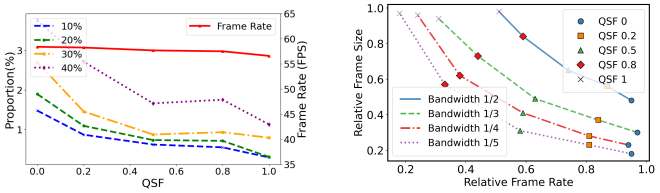
Impact of feedback delay. We dynamically change the delay between the BN and the sender to explore the impact of feedback delay. *As presented in Fig. 12, Co-RTV consistently outperforms other solutions, achieving up to 50% lower tail latency compared to FDCC and reducing delays by over 80% compared to others while maintaining high frame quality.* For the 95th frame delay, as feedback delay increases, all baselines suffer from higher feedback delay and produce uncontrollable frame delay increase, while Co-RTV maintains predictable frame delay. Though FDCC achieves the largest average frame size, the frame size gap between FDCC and Co-RTV is within 10%, offering an ultra-low latency when the network delay is up to 50 ms.

In addition, we further explored different QoE settings (τ and QSF) to verify the advancement of scalable flow control.

Impact of the empty time. As shown in Fig. 13, Co-RTV is labeled as "C" and FDCC as "F," with the numbers following each indicating the empty time values in milliseconds. Points toward the top-right indicate larger frame size and lower delay, which represents better QoE. *As the empty time τ increases, Co-RTV maintains high QoE by balancing frame size growth with minimal tail latency increase, significantly outperforming other solutions.* Notably, FDCC shows greater sensitivity to the empty time changes, as packet queuing at the BN significantly increases FDCC's tail delay when larger empty time is set. This reflects Co-RTV's ability to maintain a superior trade-off between delay and video quality.

Impact of QSF. *QSF effectively manages the trade-off between frame size and frame rate during bandwidth reduction.* To evaluate this, we recorded the overall proportion of small frames under different QSF values (Fig. 14a), as well as the variations in frame size and frame rate across different levels of bandwidth reduction and different QSF values (Fig. 14b). Specifically, we measured the proportion of frames smaller than expected, where "expected" refers to the ideal frame size based on the sending rate, defined as frames ranging between 10% and 40% of the ideal size. When QSF is set to 0, frequent queuing or frame-dropping results in a higher proportion of small frames. As QSF increases, the proportion of small frames decreases. Notably, frame skipping is only triggered when there is a significant bandwidth drop, ensuring that the overall frame rate remains largely unaffected,

[5] [71] standardizes the 5G air interface delay 4ms for eMBB, and the RTT is about 10ms.



(a) Small frame ratio (b) Frame rate and size

Fig. 14: Impact of QSF

even with QSF set to 1.

Additionally, we examined the performance of frame delay and frame size under different QSF values during a detailed bandwidth reduction event. A carefully designed bandwidth reduction experiment further demonstrated that QSF effectively balances frame size and frame rate under varying bandwidth conditions, as illustrated in Fig. 14b. In particular, in high-motion scenes where bandwidth reduction is more pronounced, increasing QSF can help safeguard video frame quality and prevent substantial degradation in visual quality.

Impact of bandwidth calculation accuracy. As Co-RTV's frame-drop control relies on the bandwidth calculation at the BN, we furthermore explore the impact of bandwidth calculation accuracy to Co-RTV in time window w. Given that achieving completely accurate bandwidth estimation is difficult in time-varying wireless networks, it is important to evaluate Co-RTV's performance when the estimation deviates from the real bandwidth.

In the emulation environment, we can obtain accurate bandwidth by directly reading the TC setting of trace $stadium$. To emulate the impact of inaccurate bandwidth calculation, we multiply the actual bandwidth by different ratios (0.8, 1.0, 1.2, 1.5) and provide them to Co-RTV. *As shown in Fig. 15, underestimation of the bandwidth results in smaller video frames, while over-estimation leads to a slight increase in both frame delay and frame size.* Notably, even a 50% over-estimation yields an overall delay difference of less than 15%. The reduced frame size observed under under-estimation conditions is attributable to Co-RTV's increased propensity for frame-drop, which causes the sender to frequently receive feedback containing lower-than-actual bandwidth, thereby producing smaller frames. Conversely, over-estimation does not substantially increase frame delay because Co-RTV controls tail latency through frame dropping and queue length feedback. As a result, Co-RTV keeps delay increases within a predictable range.

Impact of α. The parameter α functions as a frame quality sensitivity coefficient for frame-drop. As α increases, co-RTV assigns greater importance to the cost associated with quality degradation resulting from frame-drop. To investigate this relationship, we examined how the 95th percentile of frame delay and the average frame size vary across different α values. Our results reveal that decreasing α leads to reductions in both the 95th frame delay and the average video frame size. Conversely, increasing α results in pronounced increases in the 95th percentile frame delay and the average frame size.

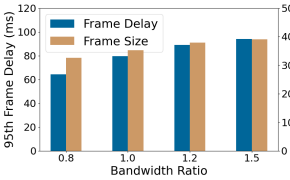


Fig. 15: Inaccurate bandwidth

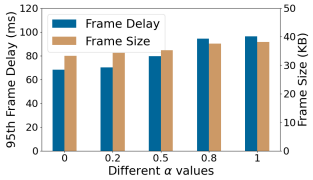


Fig. 16: Impact of α

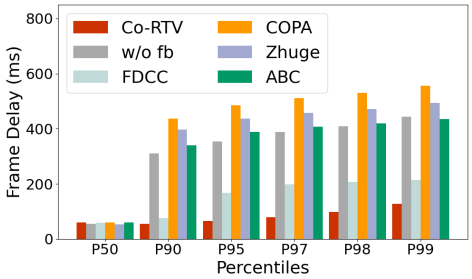


Fig. 17: 5G testbed results

C. 5G testbed Experiments

In our 5G testbed, the test RTV client was the only connection to the test CPE, ensuring that there was no interference from other streams. We conducted A/B tests for over ten thousand minutes over tens of days, alternating between six different RTV solutions, each for five-minute intervals. The logs from both the client and the server were synchronized to calculate the frame delay for each solution.

The results, shown in Fig. 17, demonstrate that Co-RTV delivers consistent low latency, outperforming other solutions. In over 80% of the cases, the video playback delay remained below 100ms, ensuring an excellent user experience. Tail latency, in particular, showed remarkable differences across solutions. Although FDCC performs reasonably well, it exhibits a higher tail latency compared to Co-RTV. In contrast, Co-RTV maintains the 99th percentile latency close to 100ms, significantly reducing 70.5% $\sim$ 77.0% of the tail latency compared to other solutions. Even for FDCC, Co-RTV achieves a 40.3% reduction of the tail latency (i.e., the 99th percentile delay).

D. Co-RTV overhead

Co-RTV incurs minimal bandwidth overhead, approximately 1% as well as minimal computational overhead. The primary source of overhead in Co-RTV, compared to other solutions, comes from the additional feedback information embedded in packet headers. In our experiments that involved tens of thousands of minutes of RTV video playback, Co-RTV's feedback accounted for only 1.3% of the total transmitted data. We further observed that Co-RTV introduces minimal computational overhead, as the base station's CPU utilization remained largely unchanged after deployment. Notably, our implementation on a Raspberry Pi 5 confirmed that Co-RTV efficiently supports real-time RTV streams at 100 Mbps, underscoring its practicality on resource-constrained Wi-Fi APs.

E. Evaluation Conclusion

Overall, the collaborative design of Co-RTV achieves higher RTV video quality and lower RTV latency. The improved video quality primarily stems from BN's feedback of queue length to the sender, enabling it to avoid unnecessary bitrate reductions in response to non-congestion events, a common issue in real-world Wi-Fi and 5G networks due to frequent forwarding delay fluctuations. Moreover, by allowing BN to promptly drop oversized video frames that cannot be forwarded in time and tell the sender to re-encode them into smaller frames, Co-RTV significantly reduces tail latency. We also include a PowerPoint presentation in the supplementary materials, featuring two videos obtained from our emulated experiments: one demonstrating Co-RTV and the other showcasing the original RTV system using COPA.

VI. Discussion and Future Work

Co-RTV deployability: Co-RTV requires moderate modifications to existing BNs (5G base stations, Wi-Fi APs) and senders. As the IETF MoQ standard continues to evolve [23], [24], such collaborative mechanisms are expected to receive broader support. Latency-sensitive RTV applications, including AR/VR, cloud gaming, and cloud-to-vehicle video, continue to face commercial challenges due to persistent latency issues [7], [8], [30], [72]–[74], underscoring the necessity of standardized collaboration between operators and OTT (Over the top) providers.

Uplink support: In uplink scenarios such as vehicle-to-cloud video, Co-RTV remains effective: mobile devices can access real-time wireless channel and queue metrics, enabling prompt feedback to RTV applications and facilitating efficient frame dropping and collaborative control, thus improving uplink performance, like PBE-CC [47].

The impact of multiple users' flow: Fortunately, in emerging networks such as 5G [32]–[36], Wi-Fi 6 [37]–[39], and Wi-Fi 7 [40]–[42], multi-user assurance is provided by forwarding their packets simultaneously. As a result, bursty traffic from other users does not significantly increase the queue length of the Co-RTV flow. Additionally, Co-RTV ensures fairness in multi users' competition scenarios as fair capacity allocation in Wi-Fi and 5G networks.

The impact of one user's multiple flows: In scenarios where a user is simultaneously receiving multiple flows, such as downloading a file while watching an RTV video, cross-stream interference can degrade performance. In practice, users often prioritize the RTV stream when experiencing stuttering by reducing or halting the file download. Future work will focus on expanding Co-RTV's capabilities to handle such multi-stream scenarios. For the same user's multiple RTV streams, the BN can evenly allocate the available bandwidth and queue length among all RTV streams. This approach ensures both fairness and low latency across multiple RTV flows for the same user.

Implementing Co-RTV to Wi-Fi AP: In Wi-Fi 6 and Wi-Fi 7 access points, Co-RTV can be easily implemented. We will conduct customized development based on the open-source OpenWrt [75] codebase and plan to release our modifications in the future. However, earlier Wi-Fi standards, such as Wi-Fi 5, utilize shared forwarding queues among users, which presents additional challenges for performance optimization.

Dynamic adaptation flow control: Real-time video content exhibits temporal variability in motion characteristics [76]; consequently, fixed values of the parameters τ and QSF may become suboptimal. We will develop an adaptive control mechanism that continuously estimates motion dynamics and updates τ and QSF to preserve the intended trade-off among perceptual quality, latency, and bitrate, which will be pursued in future work.

VII. Related Work

E2E Low Latency Optimization: Several approaches [8]–[12] focus on minimizing latency by precise E2E congestion detection mechanisms. Salsify [27] enhances this process by dynamically adjusting bitrate and reference frame encoding, enabling swift queue clearance at the sender.Moreover, techniques like Forward Error Correction (FEC) [7], [77]–[80] are commonly employed to mitigate packet loss in mobile networks, further optimizing latency. Some research also explores adaptive layered encoding strategies [81]–[83] or intelligent encoding solutions [84], [85] that allow for flexible bitrate adjustments or packet loss during network fluctuations, contributing to reduced E2E latency. Additionally, some works [86], [87] utilize multipath to optimize RTV latency. Different from these methods, Co-RTV directly drops existing queued frames to control latency proactively.

Endpoint and Network Collaboration for RTV: Solutions such as Zhuge [6] reduce the control loop by introducing delayed ACKs at the AP, allowing the sender to detect network congestion quickly. ABC [20] improves on this by feeding back acceleration or deceleration signals to the sender. PBECC [47] takes a different approach by monitoring wireless channel conditions on the user side and transmitting this information back to the sender. APN [65] introduces application-specific labels at the sender, enabling the network to adjust forwarding strategies. Compared to these approaches, Co-RTV employs explicit frame-level collaboration between the sender and the BN, enabling precise and predictable latency control.

VIII. Conclusion

We introduce Co-RTV, a system addressing high latency in RTV streaming through collaborative flow control and frame-level latency management. Through the BN's predictable frame-level latency control and the sender's scalable QoE-driven flow control, Co-RTV achieves consistently low latency. Extensive evaluations in both emulated networks and a 5G testbed demonstrate the superior performance of Co-RTV, with tail latency reductions of 69.1% and 70.5%, respectively.

Acknowledgements. We thank anonymous reviewers and shepherd for their constructive feedback. This work was supported in part by the National Key R&D Program of China (2022YFB2901800).

REFERENCES

[1] M. Torres Vega, C. Liaskos, S. Abadal, E. Papapetrou, A. Jain, B. Mouhouche, G. Kalem, S. Ergüt, M. Mach, T. Sabol *et al.*, "Immersive interconnected virtual and augmented reality: A 5g and iot perspective," *Journal of Network and Systems Management*, vol. 28, pp. 796–826, 2020.

[2] J. L. Rubio-Tamayo, M. Gertrudix Barrio, and F. García García, "Immersive environments and virtual reality: Systematic review and advances in communication, interaction and simulation," *Multimodal technologies and interaction*, vol. 1, no. 4, p. 21, 2017.

[3] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui, "A measurement study on achieving imperceptible latency in mobile cloud gaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 88–99.

[4] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2629–2642, 2017.

[5] V. Singh, A. A. Lozano, and J. Ott, "Performance analysis of receive-side real-time congestion control for webrtc," in *2013 20th International Packet Video Workshop*. IEEE, 2013, pp. 1–8.

[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] Z. Meng, X. Kong, J. Chen, B. Wang, M. Xu, R. Han, H. Liu, V. Arun, H. Hu, and X. Wei, "Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 907–926.

[8] S. Wang, S. Yang, X. Kong, C. Wu, L. Jiang, C. Xu, C. Zhao, X. Yang, J. Xiao, X. Liu *et al.*, "Pudica: Toward {Near-Zero} queuing delay in congestion control for cloud gaming," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 113–129.

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

[10] D. Ray, C. Smith, T. Wei, D. Chu, and S. Seshan, "Sqp: Congestion control for low-latency interactive video streaming," *arXiv preprint arXiv:2207.11857*, 2022.

[11] 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 7th International Conference on Multimedia Systems*, 2016, pp. 1–12.

[12] Y. Ma, H. Tian, X. Liao, J. Zhang, W. Wang, K. Chen, and X. Jin, "Multi-objective congestion control," in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 218–235. [Online]. Available: https://doi.org/10.1145/3492321.3519593

[13] M. Moulay and V. Mancuso, "Experimental performance evaluation of webrtc video services over mobile networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 541–546.

[14] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption," 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*, 2020, pp. 479–494.

[15] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, Y. Liu, F. Qian, and Z.-L. Zhang, "A first look at commercial 5g performance on smartphones," in *Proceedings of The Web Conference 2020*, 2020, pp. 894–905.

[16] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao *et al.*, "A variegated look at 5g in the wild: performance, power, and qoe implications," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 610–625.

[17] J. Son, Y. Sanchez, C. Hellge, and T. Schierl, "Adaptable l4s congestion control for cloud-based real-time streaming over 5g," *IEEE Open Journal of Signal Processing*, 2024.

[18] W. Yang, W. Du, B. Zhao, Y. Ren, J. Sun, and X. Zhou, "Cross-layer assisted early congestion control for cloud vr applications in 5g edge networks," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 2024, pp. 1–6.

[19] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ecn) to ip," Tech. Rep., 2001.

[20] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "{ABC}: A simple explicit congestion controller for wireless networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 353–372.

[21] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-l4s-arch-20, Aug. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tsvwg-l4s-arch/20/

[22] D. Brunello, I. Johansson, M. Ozger, and C. Cavdar, "Low latency low loss scalable throughput in 5g networks," in *2021 IEEE 93rd vehicular technology conference (VTC2021-Spring)*. IEEE, 2021, pp. 1–7.

[23] X. de Foy, R. Krishna, and T. Jiang, "MoQ relays for Support of High-Throughput Low-Latency Traffic in 5G," Internet Engineering Task Force, Internet-Draft draft-defoy-moq-relay-network-handling-03, Feb. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-defoy-moq-relay-network-handling/03/

[24] C. F. Jennings, S. Nandakumar, and R. Barnes, "End-to-End Secure Objects for Media over QUIC Transport," Internet Engineering Task Force, Internet-Draft draft-jennings-moq-secure-objects-02, Feb. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-jennings-moq-secure-objects/02/

[25] K. Zarifis, S. Jaiswal, I. Purushothaman, J. Varsanik, A. Tiwari, and M. Joras, "SCONEPRO Taxonomy of throttling policies used worldwide," Internet Engineering Task Force, Internet-Draft draft-zarifis-scone-taxonomy-00, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-zarifis-scone-taxonomy/00/

[26] Z. Jia, Y. Zhang, Q. Li, and X. Zhang, "Tackling bit-rate variation of rtc through frame-bursting congestion control," in *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*. IEEE, 2024, pp. 1–11.

[27] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. 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)*, 2018, pp. 267–282.

[28] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3300061.3345430

[29] NVIDIA., "Nvidia video codec sdk," https://developer.nvidia.com/video-codec-sdk, 2024.

[30] Z. Meng, T. Wang, Y. Shen, B. Wang, M. Xu, R. Han, H. Liu, V. Arun, H. Hu, and X. Wei, "Enabling high quality {Real-Time} communications with adaptive {Frame-Rate}," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1429–1450.

[31] Q. Chen and C. Li, "Argus: Real-time hq video decoding with cpu coordinating on consumer devices," in *2024 IEEE Real-Time Systems Symposium (RTSS)*, 2024, pp. 43–56.

[32] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaein, "Dynamic buffer sizing and pacing as enablers of 5g low-latency services," *IEEE transactions on mobile computing*, vol. 21, no. 3, pp. 926–939, 2020.

[33] 3GPP., "3GPP technical specification for lte." https://www.etsi.org/deliver/etsi_ts/132400_132499/132450/18.00.00_60/ts_132450v180000p.pdf, 2024.

[34] 3GPP, "System architecture for the 5g system." https://www.3gpp.org/, 2025.

[35] 3GPP, "3GPP TS 23.288: Architecture enhancements for 5g system (5gs) to support network data analytics services," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.288, December 2019, version 15.4.0. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.288/23288-f40.zip

[36] T. Zhang, J. Wang, X. S. Hu, and S. Han, "Real-time flow scheduling in industrial 5g new radio," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, 2023, pp. 371–384.

[37] K. Wang and K. Psounis, "Scheduling and resource allocation in 802.11ax," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 279–287.

[38] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on ieee 802.11 ax high efficiency wlans," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197–216, 2018.

[39] B. Bellalta, "Ieee 802.11 ax: High-efficiency wlans," *IEEE wireless communications*, vol. 23, no. 1, pp. 38–46, 2016.

[40] T. Adame, M. Carrascosa-Zamacois, and B. Bellalta, "Time-sensitive networking in ieee 802.11 be: On the way to low-latency wifi 7," *Sensors*, vol. 21, no. 15, p. 4954, 2021.

[41] ARISTA., "Wi-Fi 7: A Leap Towards Time-Sensitive Networking." https://www.arista.com/assets/data/pdf/Whitepapers/Arista-Wi-Fi-7-White-Paper.pdf, 2024.

[42] C. Deng, X. Fang, X. Han, X. Wang, L. Yan, R. He, Y. Long, and Y. Guo, "Ieee 802.11 be wi-fi 7: New challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2136–2166, 2020.

[43] W. Sentosa, B. Chandrasekaran, P. B. Godfrey, H. Hassanieh, and B. 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)*, 2023, pp. 419–436.

[44] 3GPP, "Study on scenarios and requirements for next generation access technologies," *3rd Generation Partnership Project (3GPP), Technical report (TR) 36.331*, 2017.

[45] A. Loch, I. Tejado, and J. Widmer, "Potholes ahead: Impact of transient link blockage on beam steering in practical mm-wave systems," in *European Wireless 2016; 22th European Wireless Conference*. VDE, 2016, pp. 1–6.

[46] C. Perkins, "Sending RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences," RFC 9392, Apr. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9392

[47] Y. Xie, F. Yi, and K. 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*, 2020, pp. 451–464.

[48] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9330

[49] K. D. Schepper and B. Briscoe, "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)," RFC 9331, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9331

[50] K. D. Schepper, B. Briscoe, and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)," RFC 9332, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9332

[51] X. Lin, D. Yu, and H. Wiemann, "A primer on bandwidth parts in 5g new radio," *5G and Beyond: Fundamentals and Standards*, pp. 357–370, 2021.

[52] Y. Lin, Y. Gao, and W. Dong, "Bandwidth prediction for 5g cellular networks," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 2022, pp. 1–10.

[53] D. Xu, A. Zhou, G. Wang, H. Zhang, X. Li, J. Pei, and H. 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*, 2022, pp. 729–742.

[54] W. Ye, X. Hu, S. Sleder, A. Zhang, U. K. Dayalan, A. Hassan, R. A. Fezeu, A. Jajoo, M. Lee, E. Ramadan *et al.*, "Dissecting carrier aggregation in 5g networks: Measurement, qoe implications and prediction," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 340–357.

[55] 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.

[56] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee, "Exll: An extremely low-latency congestion control for mobile cellular networks," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, 2018, pp. 307–319.

[57] A. Nota, S. Saidi, D. Overbeck, F. Kurtz, and C. Wietfeld, "Context-based latency guarantees considering channel degradation in 5g network slicing," in *2022 IEEE Real-Time Systems Symposium (RTSS)*, 2022, pp. 253–265.

[58] J. Zhang, F. Yang, T. Liu, Q. Wu, W. Zhao, Y. Zhang, W. Chen, Y. Liu, H. Guo, Y. Ma *et al.*, "{TECC}: Towards efficient {QUIC} tunneling via collaborative transmission control," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 253–266.

[59] M. Palmer, M. Appel, K. Spiteri, B. Chandrasekaran, A. Feldmann, and R. K. Sitaraman, "Voxel: Cross-layer optimization for video streaming with imperfect transmission," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 359–374.

[60] OpenAirInterface, "Openairinterface 5g: Feature set documentation," https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/FEATURE_SET.md, accessed: 2024-10-09.

[61] F. Kaltenberger, G. De Souza, R. Knopp, and H. Wang, "The openair-interface 5g new radio implementation: Current status and roadmap," in *WSA 2019; 23rd International ITG Workshop on Smart Antennas*. VDE, 2019, pp. 1–5.

[62] A. online, "Oai-ran." [Online]. Available: https://openairinterface.org/oai-5g-ran-project/

[63] WITCOMM., "xgproduct." https://witcomm.net/xgstation, 2023.

[64] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002, pp. 89–102.

[65] Z. Li, S. Peng, C. Xie, and S. Zhang, "Application-aware IPv6 Networking (APN6) Encapsulation," Internet Engineering Task Force, Internet-Draft draft-li-6man-apn-ipv6-encap-00, Mar. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-li-6man-apn-ipv6-encap/00/

[66] M. Scharf and S. Kiesel, "Nxg03-5: Head-of-line blocking in tcp and sctp: Analysis and measurements," in *IEEE Globecom 2006*. IEEE, 2006, pp. 1–5.

[67] Alibaba., "XQUIC Library released by Alibaba is a cross-platform implementation of QUIC and HTTP/3 protocol." https://github.com/alibaba/xquic, 2022.

[68] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 183–196.

[69] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[70] I. Johansson and Z. Sarker, "Self-clocked rate adaptation for multimedia," Tech. Rep., 2017.

[71] 3GPP., "TR 138 913 - V18.0.0 - 5G; Study on scenarios and requirements for next generation access technologies (3GPP TR 38.913 version 18.0.0 Release 18)," https://www.etsi.org/deliver/etsi_tr/138900_138999/138913/18.00.00_60/tr_138913v180000p.pdf, 2024.

[72] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE network*, vol. 32, no. 2, pp. 78–84, 2018.

[73] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5g ran slicing for verticals: Enablers and challenges," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 28–34, 2019.

[74] Y. Zhang, M. Chen, N. Guizani, D. Wu, and V. C. Leung, "Sovcan: Safety-oriented vehicular controller area network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 94–99, 2017.

[75] O. Project., "Open wireless router, an open-source project for embedded operating systems based on Linux, primarily used on embedded devices to route network traffic." https://openwrt.org/, 2024.

[76] Y. Zhao, F. Yang, G. Lv, Q. Wu, Y. Liu, J. Zhang, Y. Peng, F. Peng, H. Guo, Y. Chen *et al.*, "{MARC}:{Motion-Aware} rate control for mobile e-commerce cloud rendering," in *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, 2025, pp. 217–232.

[77] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward error correction (fec) building block," Tech. Rep., 2002.

[78] K. Park and W. Wang, "Afec: An adaptive forward error correction protocol for end-to-end transport of real-time traffic," in *Proceedings 7th International Conference on Computer Communications and Networks (Cat. No. 98EX226)*. IEEE, 1998, pp. 196–205.

[79] L. Vicisano, M. Watson, and M. Luby, "Forward Error Correction (FEC) Building Block," RFC 5052, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc5052

[80] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 953–971.

[81] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[82] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[83] M. Dasari, K. Kahatapitiya, S. R. Das, A. Balasubramanian, and D. Samaras, "Swift: Adaptive video streaming with layered neural codecs," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 103–118.

[84] Y. Cheng, Z. Zhang, H. Li, A. Arapin, Y. Zhang, Q. Zhang, Y. Liu, K. Du, X. Zhang, F. Y. Yan *et al.*, "{GRACE}:{Loss-Resilient}{Real-Time} video through neural codecs," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 509–531.

[85] Y. Cheng, A. Arapin, Z. Zhang, Q. Zhang, H. Li, N. Feamster, and J. Jiang, "Grace++: Loss-resilient real-time video communication under high network latency," *arXiv preprint arXiv:2305.12333*, 2023.

[86] Y. Zhou, T. Wang, L. Wang, N. Wen, R. Han, J. Wang, C. Wu, J. Chen, L. Jiang, S. 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)*, 2024, pp. 1901–1916.

[87] 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.