Breath: Adaptive Protection Boundary in FEC Encoding for Mobile Real-Time Video Streaming

Shiyang Huang*
huangshiyang23s@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Gerui Lv*
lvgerui@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Yuankang Zhao
zhaoyuankang@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Jiaxing Zhang
zhangjiaxing20g@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Qingyue Tan
tanqingyue22s@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences,
Beijing, China

Congkai An
ACK@bupt.edu.cn
Beijing University of Posts and Telecommunications,
Beijing, China

Huanhuan Zhang
zhanghuanhuan@bupt.edu.cn
Beijing University of Posts and Telecommunications,
Beijing, China

Xinyi Zhang
xyzhang@cnic.cn
Computer Network Information Center,
Chinese Academy of Sciences, Beijing, China

Qinghua Wu†
wuqinghua@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Zhenyu Li
zyli@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences,
Beijing, China

Abstract

Mobile real-time video streaming (RTVS) demands ultra-low latency to preserve content timeliness. Packet loss in mobile networks significantly inflates frame latency and thus degrades the quality of experience (QoE). As a promising solution, Forward Error Correction (FEC) encoding has been widely deployed in RTVS systems to recover from packet loss by introducing redundancy. However, existing schemes focus on per-frame FEC protection, failing to optimize QoE because they cannot precisely allocate redundancy to handle burst loss events. These events typically occur at the single-frame level, but can be smoothed out at the multi-frame level. We propose Breath, an adaptive FEC scheme that dynamically adjusts the protection boundary based on network and video dynamics. We have implemented Breath in a RTVS system and evaluated it in emulated mobile networks using network traces collected from the production system. Results show that, compared to state-of-the-art FEC schemes, Breath reduces deadline missing rate by 17.2%-22.5% while improving the average video bitrate by 10.6%-14.2%.

CCS Concepts

• **Networks → Cross-layer protocols**.

Keywords

Real-time communications, Forward error correction, Quality of experience

ACM Reference Format:
Shiyang Huang, Gerui Lv, Yuankang Zhao, Jiaxing Zhang, Qingyue Tan, Congkai An, Huanhuan Zhang, Xinyi Zhang, Qinghua Wu, and Zhenyu Li. 2026. Breath: Adaptive Protection Boundary in FEC Encoding for Mobile Real-Time Video Streaming. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3774904.3792265

1 Introduction

Mobile Real-Time Video Streaming (RTVS) applications, such as video conferencing [29], live streaming [25, 27, 51], and cloud gaming [20, 40, 49], impose strict end-to-end (E2E) latency constraints (often below 150 ms [18, 30, 31, 38, 43, 52]) to maintain user interactivity [11, 53]. However, meeting such latency requirements is challenging because empirical studies like [15] identify packet loss as a primary cause of latency inflation, especially in mobile networks [1, 12]. Existing loss recovery solutions [9, 26, 34] that rely

[*] Co-first authors.
[†] Corresponding author.

This work is licensed under a Creative Commons Attribution 4.0 International License.

WWW '26, Dubai, United Arab Emirates.
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792265

on retransmissions incur a latency penalty of at least one additional Round-Trip Time (RTT), frequently causing the retransmitted data to arrive too late and breach the strict 150 ms deadline.

In contrast to reactive retransmission-based solutions, Forward Error Correction (FEC) [28, 41, 48] provides a proactive mechanism for loss recovery that eliminates the additional latency introduced by retransmissions. By proactively generating redundant packets from a block (*i.e.,* protection boundary) of source media data, FEC enables the receiver to reconstruct lost packets by using redundant packets alongside the original media packets, without waiting for sender retransmission. In modern FEC for RTVS, protection is applied at the per-frame level. Consequently, mainstream FEC schemes aim to carefully allocate an appropriate number of redundant packets to each frame based on loss estimation, ensuring efficient loss recovery while limiting the bandwidth wasted by unnecessary redundancy [2, 4, 6–8, 23, 38].

Despite the above efforts, our large-scale measurements from over 65,000 sessions on a mobile RTVS system reveal that, although *existing FEC schemes provide sufficient redundancy at the session level, they cannot effectively protect frames against severe frame-level loss events.* Surprisingly, an additional 9.5% redundancy results in 27.6% of frames being unable to recover. The root cause is that existing solutions fail to precisely allocate the necessary redundant resources to recover from *burst loss events.* Even worse, these events mostly (in about 80% of cases) occur in a single frame, rather than continuously and evenly across frames, making them almost unpredictable.

Fortunately, we find it possible to utilize the burst nature of per-frame loss events to improve FEC protection efficiency without optimizing loss estimation methods. Specifically, a frame that has experienced severe loss is often surrounded by multiple loss-free or low-loss frames. Therefore, expanding the FEC protection boundary to encompass these surrounding frames provides an opportunity to transfer unused redundancy from them to the frame experiencing high burst loss. Based on this insight, we reevaluate the effectiveness of per-frame protection and *expand the design space by applying FEC encoding to multiple frames.* Although the core idea is feasible, implementing it in a practical mobile RTVS system presents several challenges:

***(i)* Adapting to complex system dynamics.** The effectiveness of the FEC scheme depends on various dynamic factors, including network conditions (e.g., loss rate and RTT) and video features (e.g., frame size). In RTVS, these factors exhibit high volatility, frequently fluctuating within a single frame interval, and even more so across multiple frames. Therefore, setting a fixed protection boundary is not enough, posing a challenge in continuously determining the optimal decision to adapt to these dynamics.

***(ii)* Fine-grained and swift decision-making.** The system needs to decide how many frames should be included in a protection boundary in real time. This requires fine-grained decision-making, *e.g.,* at the frame level. However, the trend toward higher frame rates (e.g., 120 fps) reduces the decision time window to a few milliseconds (e.g., 8.3 ms). Thus, the FEC scheme must be computationally efficient to operate under such stringent time constraints.

To address the above challenges, we propose Breath, an **adaptive protection boundary** mechanism for FEC encoding in mobile RTVS. Breath aims to optimize the Quality of Experience (QoE) for users, *i.e.,* reducing latency by FEC protection while maintaining video quality. Although extending the FEC protection boundary improves the frame protection efficiency, it also introduces additional overhead that harms QoE performance (Sec. 2.3). To this end, Breath incorporates a QoE-driven mathematical model (Sec. 3.2) that quantifies the FEC protection overhead and further solves the overhead minimization problem online (Sec. 3.3). To enable swift, frame-level decision-making, Breath adopts a response-based method to dynamically construct the protection block, thereby limiting the impact of prediction uncertainty.

We have implemented Breath in a practical RTVS system and extensively evaluated its effectiveness through trace-driven evaluations using network traces collected in our production system (Sec. 2.2). Experimental results demonstrate that compared to state-of-the-art FEC schemes, Breath reduces the Deadline Missing Rate (DMR) [31] by 17.2%-22.5% while improving the average video bitrate by 10.6%-14.2%, successfully pushing forward the Pareto frontier of FEC protection efficiency and overhead.

In summary, this paper makes the following contributions:

- Identifying the fundamental challenge faced by existing FEC solutions through a large-scale measurement study on a mobile RTVS production system (Sec. 2).
- Proposing Breath that extends the design space of FEC encoding by adaptively deciding the protection boundary (Sec. 3).
- Implementing Breath in a mobile RTVS system, ensuring real-time decision-making under strict time constraints (Sec. 3.4).
- Evaluating Breath's effectiveness through thorough experiments using traces collected from the production system (Sec. 4).

2 Background and Motivation

2.1 FEC in Mobile Real-Time Video Streaming

In RTVS systems, the sender typically encodes video content into a series of video frames, which are further packaged into media packets and sent to the receiver for playout. The loss of any media packet belonging to a video frame can lead to severe latency increase and quality degradation, such as visual corruption and playback freezing. Consequently, to provide a smooth viewing experience, all media packets for a given frame should be received in their entirety.

However, maintaining the timely delivery of video frames in a lossy link (*i.e.,* in mobile networks) is challenging [37]. As a solution, FEC is widely applied in RTVS to protect frames from packet loss. Mainstream FEC schemes generate redundant packets from a group of media packets and transmit them alongside the original media. Lost media packets can be recovered at the receiver through FEC decoding of the redundant packets, if the total number of received packets is no less than that of the original media packets.

Most existing FEC schemes typically work on the per-frame granularity, *i.e.,* treating the media packets of one frame as a protection (encoding) block. To achieve successful protection, these schemes are designed to ensure that the *redundancy rate* exceeds the *estimated loss rate* of each frame. The redundancy rate is calculated by dividing the number of redundant packets by the number of media packets. In addition, the loss rate of the next frame is estimated by filtering or smoothing methods based on past loss rate observations (**rule-driven**, *e.g.,* WebRTC [47]), or by the output of a neural network that takes the past loss rate and other status as input (**learning-driven**, *e.g.,* DeepRS [8] and Tooth [4]).

While more redundancy increases the recovery probability, it occupies the available bandwidth of video content, thus degrading video quality. To curb the bandwidth overhead, RTVS systems commonly limit the maximum redundancy rate to below a specific ratio of media packets. For example, WebRTC [47] limits it to 50%.

2.2 Limitations of Existing FEC Schemes

To investigate how the existing FEC-based solutions perform in the wild Internet, we present the results of a large-scale study measuring the RTVS service of one of the top-tier streaming platforms.

System setup. Our system is built on the open-source WebRTC [47] architecture and deploys WebRTC's flexible FEC scheme with the default settings, representing a standard solution in large-scale RTVS transmission systems. WebRTC estimates the loss rate by the maximum of the average loss rate observed in the past time windows (*e.g.,* 100 ms) and uses a built-in lookup table to map the estimated loss rate to a required redundancy rate for each frame.

Dataset. The collected dataset involves over 100,000 users and spans 65,742 video sessions that experienced packet loss events, accumulating over 153 million video frames and 1,500 hours of video streaming. It encompasses a wide range of users from various mobile and wireless networks and geographical locations and ISPs, enabling us to uncover systemic challenges of FEC in the wild. For each video session, the following performance metrics are collected: (*i*) frame size in bytes; (*ii*) available bandwidth of each frame (provided by WebRTC); (*iii*) number of lost packets during transmission of each frame; (*iv*) number of media packets and redundancy (FEC) packets contained in each frame; (*v*) transmission latency of each frame, measured from the frame being encoded at the sender to being ready for decoding at the receiver[1]. Note that all data were collected with user permission and contained no private user information. By analyzing this dataset, we found that:

Observation: **Even though existing FEC schemes provide adequate redundancy, they still cannot protect frames from frequent loss events.**

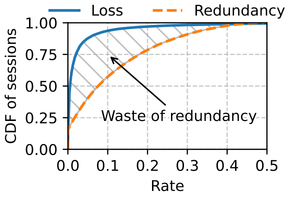

Figure 1: Session average loss and redundancy rates.

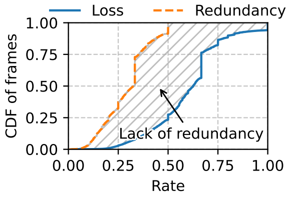

Figure 2: Loss and redundancy rates in failed frames.

Our measurements reveal a critical paradox in the performance of the deployed FEC. As shown in Fig. 1, WebRTC provides 11.2% redundancy on average across all sessions, while the average loss rate is only 2.7%. This suggests that FEC provides an **additional 9.5% of protection** for video frames. Ideally, this redundancy could cover all lost packets, or at least most of them. However, it is striking that even when protected by the FEC, **27.6%** of the frames that encountered loss events failed to be decoded (referred to as "failed frames"), as illustrated in Fig. 2. Such failed frames can only be recovered through retransmission at the sender, but this process requires at least one RTT to complete, resulting in prolonged frame latency. The result shows that the median transmission latency of failed frames is significantly (2.7 times) higher than that of recovered frames[2], reaching 236 ms (Fig. 3), which is far beyond the 150 ms transmission deadline [18, 30, 31, 38, 43, 52].

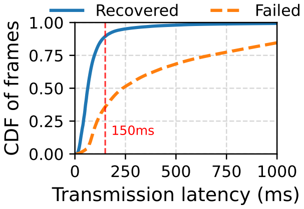

Figure 3: Latency distribution of recovered vs. failed frames.

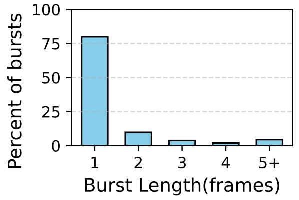

Figure 4: Burst length of high-loss events in failed frames.

Drilling down into these failed frames reveals that they suffer from extremely high packet loss: 72.5% of them experience a loss rate higher than 50%, which exceeds the maximum frame redundancy rate allowed by the system. Therefore, we further conclude that *these high-loss events are transient bursts instead of sustained periods of poor connectivity.* As shown in Fig. 4, about 80% of high-loss events (*i.e.,* loss rate >50%) last for only one frame, and over 93% span three frames. Similar observations have also been reported in recent studies like [38].

These results indicate that high-loss events have low temporal correlation, which makes them extremely difficult to foresee. Consequently, even when there is sufficient redundancy budget at the session level, existing FEC schemes face a fundamental challenge in precisely allocating redundancy to each frame for recovery.

Root Cause: **Protection failures stem from the temporal misallocation of redundant resources due to unpredictable, high-intensity burst loss events.**

For rule-driven FEC schemes like WebRTC, these burst loss events are often too short-lived to be reliably captured and accurately predicted by backward-looking loss rate predictors. Our dataset shows that a long-term statistical window may report a much lower average loss rate (*e.g.,* 7.7%), while masking the fact that these packet losses were concentrated in a single, catastrophic burst (*e.g.,* with 57% loss rate) in a frame.

Learning-driven FEC schemes also struggle with such prediction uncertainty. The features available before transmission (e.g., past network statistics) often have low correlation with future, instantaneous burst loss events, making the loss rate prediction intractable. A slight prediction error during a burst can result in an entire frame failing to be recovered.

When addressing the observed issue, FEC schemes must choose between (*i*) setting a continuously high redundancy rate, which wastes significant bandwidth on frames that will not experience loss events, and (*ii*) maintaining low redundancy, which is insufficient for protecting against burst loss.

[1] This paper focuses on transmission latency because it accounts for nearly 80% of the E2E latency in our system. Previous studies like [50, 53] have also observed that transmission latency dominates E2E latency.

[2] "Recovered frame" indicates a frame for which specific packets were lost, but were recovered by the receiver based on redundant FEC packets.

Summary. Our analysis reveals a fundamental limitation of existing FEC solutions: They rely on pre-emptive, per-frame loss estimation and redundancy allocation. This approach proves inefficient against the unpredictable and dramatic variance of per-frame loss rates, creating a dilemma between excessive bandwidth overhead and frequent recovery failures.

2.3 Opportunity: Extending the FEC Boundary across Frames

As indicated above, per-frame FEC schemes are inherently vulnerable to the unpredictable nature of the loss rate at the frame level. Thus, we have the following question: *Is it possible to improve the FEC protection efficiency without pursuing a perfect loss rate estimation method?*

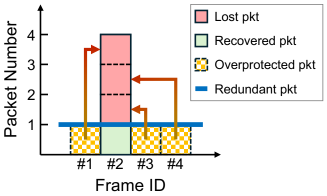

Figure 5: Idea of multi-frame FEC protection.

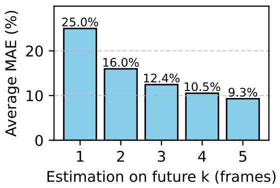

Figure 6: Estimation error at different frame levels.

Basic idea. The chance lies in our former observation that frames adjacent to the one that has experienced burst packet loss are often free of packet loss or only encounter slight loss events (Fig. 4). In other words, using additional redundancy resources in other frames to cover burst loss events in a specific frame is feasible. This finding prompts us to expand the perspective of the FEC protection from a single frame to **multiple frames**.

Fig. 5 provides a simple illustration of how this idea works[3]. Assume there are four consecutive frames, each containing the same number of media packets. Specifically, frame #2 will encounter four lost packets, while the other three frames will experience no loss. Suppose that the estimated loss packets (loss rate multiplied by media packets) is one packet per frame, corresponding to the redundant packet provided by FEC (blue line in Fig. 5). In this scenario, most FEC schemes focusing on single-frame protection will suffer from three lost packets in frame #2 (light red squares in Fig. 5) due to underestimating its loss rate. However, frames #1, #3, and #4 are *overprotected*, meaning they have redundant packets that are not used to recover lost packets. Fortunately, if the FEC scheme sets its protection boundary to all four frames, using the extra redundant packets in the loss-free frame can perfectly recover the lost packets in frame #2, as highlighted by the three arrows in Fig. 5.

Additionally, we have also observed that the burst nature of per-frame loss can be smoothed out at the multi-frame level. As shown in Fig. 6, the loss rate estimation error decreases when the observation unit expands from one frame to five frames, in terms of mean absolute error (MAE). Note that these accuracy improvements were achieved using the default loss rate estimator in WebRTC. This phenomenon can be explained by the Law of Large Numbers [14]: The observed loss rate is more likely to converge to the expected mean as the number of samples increases (as reported in [4]). These results bring the following insight:

> ***Insight:*** **Expanding the FEC protection boundary from one frame to multiple frames can reduce the impact of burst loss unpredictability.**

In essence, the issue in Sec. 2.2 can be considered a *supply-demand imbalance* problem, in which the receiver demands the redundancy supplied by the sender against loss events. Our insight indicates that mapping the FEC supply and demand is much easier at a coarser, multi-frame level than at the single-frame level due to the uneven and discontinuous nature of loss events across frames.

Strawman solution. Recall that RTVS systems should adhere to the 150 ms transmission deadline for each frame. Since FEC redundant packets are usually sent with the protected media packets, an intuitive method following our insight and Fig. 6 is to maximize the protection boundary while ensuring the protected frame range does not exceed the single-frame transmission deadline. Namely, set the protection boundary as $\lfloor (deadline\ \text{-}\ OWD) \times frame\ rate \rfloor$ frames [38]. Here, OWD indicates the one-way delay from the sender to the receiver. For example, with a frame rate of 30 fps, an OWD of 30 ms, and a deadline of 150 ms, this method (denoted as "Mpb-WebRTC") sets the protection boundary to three frames.

Our controlled experiments in Sec. 4 confirm that compared to the original WebRTC, Mpb-WebRTC indeed improves the recovery rate (*i.e.,* the ratio of recovered frames to all frames) from 91.1% to 95.2%, as shown in Fig. 9. However, Mpb-WebRTC decreases the user-perceived QoE performance in terms of latency. Tab. 2 indicates that Mpb-WebRTC performs worse than WebRTC on almost all latency metrics, including average, P50, and P95 values. This is because the generation of redundant packets starts after all frames within the block are encoded. Consequently, the receiver cannot recover the former lost frames in a block until it receives the redundant packets, which increases the latency of all recovered frames. Critically, as the protection boundary continues to expand, it yields diminishing marginal returns in recovery probability, while the recovery latency grows linearly.

To summarize, the strawman solution that adopts the maximum protection boundary improves the FEC recovery rate but fails to optimize QoE performance. Therefore, a more effective method needs to dynamically adjust the protection boundary for balancing FEC protection efficiency and overhead (*e.g.,* transmission latency), which motivates our work in this paper.

3 Design and Implementation

Based on the above, we propose Breath, an adaptive protection boundary mechanism for FEC encoding in mobile RTVS. Breath aims to optimize user QoE by utilizing adaptive boundary FEC to protect frames from loss bursts. To do so, Breath carefully controls the temporal (*i.e.,* recovery latency) and spatial overhead (*i.e.,* redundant packets) introduced by FEC, in order to minimize frame latency without compromising video bitrate.

3.1 Design Challenges and Solutions

Challenge 1: Intrinsic trade-offs in spatial-temporal joint optimization. In the temporal dimension, extending the protection boundary improves recovery probability but introduces additional recovery latency. In the spatial dimension, increasing redundancy also enhances recovery probability but consumes effective bitrate. Since these two decisions are tightly coupled, jointly affecting QoE,

[3] This example only showcases the core concept and has been simplified for easier understanding. The practical implementation is more complicated (details in Sec. 3).

Table 1: Notations defined in Breath Design

Notation	Meaning
Inputs:	
I	Frame interval (ms)
$d(i)$	Number of packets in frame i
lr	Average Loss rate (%) in a protection block
sr_i	Sending rate (Bytes/ms) of frame i
OWD_i	Average OWD (ms) in frame i
RTT_i	Average RTT (ms) in frame i
MTU	Maximum transmission unit (Bytes)
Intermediate variables:	
M	Number of packets in a protection block
R	Redundancy rate (%) in a protection block
rtx_i	Retransmit round(s) to complete transmitting frame i
$P^{(i)}_{\text{recover}}$	Probability of frame i succeeding to be recovered
$P^{(i)}_{\text{fail}}$	Probability of frame i failing to be recovered
L_i	Transmission latency of frame i
$L^{(i)}_{\text{recover}}$	Latency of frame i succeeding to be recovered
$L^{(i)}_{\text{fail}}$	Latency of frame i fails to be recovered
Parameters:	
λ	Weight for redundant bandwidth cost
ω	Weight for FEC protection failure case
Outputs:	
N	Number of frames in a protection block
red	Number of redundant packets in a protection block

the key challenge lies in quantifying and jointly optimizing latency and redundancy overhead within a unified framework.

Solution: A QoE-driven FEC overhead model (Sec. 3.2). We develop a mathematical model that formulates the expected temporal and spatial overhead of any FEC protection scheme. The model quantifies the expected benefit in recovery rate improvement and the corresponding temporal cost in recovery latency, formulates their effect on video frame transmission latency and integrates it with the spatial cost of redundancy rate.

Challenge 2: Computational complexity in FEC decisions. Determining both the protection boundary and the redundancy rate constitutes a high-dimensional problem. Achieving a globally optimal solution within the millisecond-scale decision window required by streaming applications is computationally difficult, posing a severe challenge to real-time operation.

Solution: Lightweight response-based decision (Sec. 3.3). Breath incorporates a lightweight response-based decision mechanism. For each newly encoded video frame, Breath performs a swift expectation comparison between two options: (*i*) finish the current protection block or (*ii*) extend it to the next frame. This binary decision mechanism compresses the search space from a continuous high-dimensional domain to a discrete binary one, achieving efficient real-time adaptation.

3.2 Modeling FEC Overhead

We begin by presenting a mathematical model that quantifies the FEC protection overhead in RTVS. This model provides a universal framework to evaluate the effectiveness of mainstream FEC schemes. Tab. 1 summarizes the key variables in our model. We model the expected overhead of an FEC protection boundary across N frames with red redundant packets as shown in Eq. (1) to Eq. (3).

$$\mathbb{E}_{\text{overhead}}(N, red) = \sum_{i=1}^{N} \mathbb{E}_{L_i}(N, red) + \lambda \cdot R(N, red), \tag{1}$$

where $\sum_{i=1}^{N} \mathbb{E}_{L_i}(N, red)$ and $\mathrm{R}(N, red)$ respectively denote the temporal and spatial overhead. The joint optimization of these two overheads can be modeled in multiple ways. We adopt a linear combination of them to be the expected overhead for computational convenience, but our framework can be applied to alternative modeling approaches. $\mathbb{E}_{L_i}$can be further expressed by Eq. (2):

$$\mathbb{E}_{L_i} = P^{(i)}_{\text{recover}}(N, red) \cdot L^{(i)}_{\text{recover}}(N, red) + \omega \ \cdot P^{(i)}_{\text{fail}}(N, red) \cdot L^{(i)}_{\text{fail}}(N, red), \tag{2}$$

where $P^{(i)}_{recover}$ and $P^{(i)}_{fail}$ represent the probability of the i-th frame in the protection block being successfully recovered or failing to be recovered. $L^{(i)}_{recover}$ and $L^{(i)}_{fail}$ refer to the transmission latency of the i-th frame to be recovered by FEC decoding or by retransmission when the FEC recovery fails. Spatial overhead R is defined in Eq. (3), also referred to as the redundancy rate.

$$R = \frac{red}{\sum_{i=1}^{N} d(i)}. \tag{3}$$

We introduce ω and λ to respectively control user preference for better tail latency performance and less redundant bandwidth cost. By tuning these parameters, system designers can adapt the model's behavior to specific application requirements.

Through the model, we reflect the impact of FEC on user QoE. A lower overhead indicates that lost frames are more likely to be recovered quickly with less redundancy, resulting in better user QoE in terms of higher video quality and shorter frame latencies.

Recovery probability and expected latency formulation. Next, we examine how different FEC protection boundaries and redundancy rates influence the FEC overhead. We formulate the probability and expected transmission frame latency of each recovery case, taking into account multiple factors including video frame size, network status and retransmission.

Considering an FEC protection block spanning N frames and protected by red redundant packets, let M denote the total number of packets in the protection block defined by Eq. (4).

$$M = \sum_{i=1}^{N} d(i) + red. \tag{4}$$

When video frames in the block are transmitted over a network link with a loss rate of lr, they can be classified into three cases: **Lossless**, **FEC Recovery**, and **RTX Recovery**. For each case, we formulate the expected transmission latency and probability of every frame in the protection block.

If a frame is received without any packet loss, neither FEC nor retransmission is triggered. Since this case does not involve any loss recovery mechanism, it is excluded from further analysis.

The second case refers to a frame that experiences packet loss but can be successfully recovered by FEC at the receiver. The probability of this case is given by Eq. (5).

$$P^{(i)}_{\text{recover}} = \sum_{j=1}^{red} \binom{M}{j} (lr)^j (1-lr)^{M-j} - \sum_{j=1}^{red} \binom{M-d(i)}{j} (lr)^j (1-lr)^{M-j}. \tag{5}$$

In this case, the expected frame transmission latency equals the sum of the protection block transmission time and the OWD of the frame, as defined by Eq. (6).

$$L^{(i)}_{\text{recover}} = \begin{cases} (N-i) \cdot I + \dfrac{(d(N)+red) \cdot MTU}{sr_i} + OWD_i, & \eta = 1, \\ \min\Big\{ (N-i) \cdot I + \dfrac{(d(N)+red) \cdot MTU}{sr_i} + OWD_i, & \\ \quad \dfrac{d(i) \cdot MTU}{sr_i} + OWD_i + RTT_i \cdot \mathbb{E}[rtx_i] \Big\}, & \eta = 0. \end{cases} \tag{6}$$

where $\eta = 1/0$ defines whether FEC-coupled retransmission[4] is used. rtx_i is a random variable dependent on $d(i)$ and lr and its expectation can be calculated by Eq. (7) as shown below:

$$\mathbb{E}[rtx_i] = \sum_{k=1}^{\tau}\left[1 - \left(1 - lr^{k}\right)^{d(i)}\right] = \sum_{j=1}^{d(i)} (-1)^{j+1}\binom{d(i)}{j}\frac{lr^{j}\left(1 - (lr^{j})^{\tau}\right)}{1 - lr^{j}}. \tag{7}$$

where τ denotes the maximum retransmission rounds allowed for a single packet by the system.

The third case occurs when the FEC block experiences excessive packet losses and fails to recover any lost frames. In this case, the lost frames are recovered via retransmission. The transmission latency of these frames equals the transmission time of either the frame itself or the entire protection block plus the OWD and RTT multiplied by retransmission rounds, as defined in Eq. (8). The probability of this case, denoted as $P_{\text{fail}}(i)$, is given in Eq. (9). For cases involving encoder overshoot [13, 16, 53, 54], please refer to Appendix C for the adjusted latency calculation.

$$L_{\text{fail}}^{(i)} = \begin{cases} (N-i)\cdot I + \dfrac{(d(N)+red)\cdot MTU}{sr_i} + OWD_i + RTT_i \cdot \mathbb{E}[rtx_i], & \eta = 1 \\ \frac{d(i)\cdot MTU}{sr_i} + OWD_i + RTT_i \cdot \mathbb{E}[rtx_i], & \eta = 0 \end{cases} \tag{8}$$

$$P_{\text{fail}}^{(i)} = \sum_{j=red+1}^{M}\binom{M}{j}(lr)^{j}(1-lr)^{M-j} - \sum_{j=red+1}^{M-d(i)}\binom{M-d(i)}{j}(lr)^{j}(1-lr)^{M-j}. \tag{9}$$

The transmission latency of the third case is the highest and represents the main contributor to the tail latency among all frames, which directly impacts QoE. Therefore, we introduce ω in Eq. (2) to assign a higher weight to this case, encouraging the FEC scheme to reduce the chance of FEC failure and to optimize the tail latency.

3.3 Breath Design

Overhead minimization problem. Based on the FEC overhead model in Sec. 3.2, we can formulate the FEC protection boundary and redundancy rate control problem as a constrained optimization problem of minimizing the expected FEC overhead:

$$\min_{N, red} \mathbb{E}_{\text{overhead}} \tag{10}$$
$$s.t. \text{ Equation (1) to (9).}$$

Searching for an optimal protection boundary and redundancy rate for a series of future frames in advance is highly susceptible to network fluctuations. The inherent inaccuracy of network state prediction can significantly degrade the performance of this method. Therefore, Breath is designed to operate in a response-based manner. As we will detail below, this approach not only allows for better adaptation to network dynamics but also significantly reduces the decision search space.

Specifically, we define a FEC decision as a pair (pb, red), where $pb \in \{0, 1\}$ is a binary flag indicating whether to finish the protection block on the current frame. Alg. 1 shows how Breath determines the optimal (N, red) using pb.

Each time a new frame is encoded, Breath appends its packets to the ongoing protection block, incrementing the accumulated frame count N by one. It then traverses from red^* (red from the last round) to possible redundancy levels (e.g., up to 100% of media packet numbers) to determine the red that yields the minimum current FEC overhead, denoted as $(\mathbb{E}_{\text{overhead}})_{\text{now}}$. It then utilizes the predicted next frame size $d(\hat{N+1})$ derived from the latest target video bitrate to calculate the minimum overhead of extending the protection boundary to the next frame, denoted as $(\mathbb{E}_{\text{overhead}})_{\text{next}}$. Then, Breath compares the two. If $(\mathbb{E}_{\text{overhead}})_{\text{next}}$ is smaller, set $pb = 0$ and hold the FEC protection block unfinished until the next frame is encoded; otherwise, set $pb = 1$, and red redundant packets are generated and transmitted.

This allows Breath to remain responsive to network dynamics by making timely decision updates. It also reduces the search space to at most $2 \times red_{\max}$, where $red_{\max}$ denotes the maximum number of redundant packets per block, enabling Breath to operate efficiently within each frame interval (see Appendix E for detailed computational overhead analysis).

Algorithm 1 Breath: Adaptive Protection Boundary FEC

```
Input: $d(1), \ldots, d(N), d(\hat{N+1}), red^*, lr, I, sr_i, RTT_i, OWD_i$
parameters: $\omega, \lambda$
Output: $pb, red$
// Step 1: Compute minimal overhead and optimal $red$ with current $N$ frames (up to 100% redundancy).

$(\mathbb{E}_{\text{overhead}})_{\text{now}} \leftarrow \min_{red \in [red^*, \sum_{i=1}^{N} d(i)]} \mathbb{E}_{\text{overhead}}(N, red)$
$red_{\text{now}} \leftarrow \arg\min_{red} \mathbb{E}_{\text{overhead}}(N, red)$
// Step 2: Compute minimal overhead with predicted $N+1$ frames

$(\mathbb{E}_{\text{overhead}})_{\text{next}} \leftarrow \min_{red \in [red^*, \sum_{i=1}^{N+1} d(i)]} \mathbb{E}_{\text{overhead}}(N+1, red)$
// Step 3: Make protection boundary decision

if $(\mathbb{E}_{\text{overhead}})_{\text{next}} < (\mathbb{E}_{\text{overhead}})_{\text{now}}$ then
    $pb \leftarrow 0, red^* \leftarrow red_{\text{now}}$                ▷ Hold block
else
    $pb \leftarrow 1, red \leftarrow red_{\text{now}}$                  ▷ Finish block
end if
```

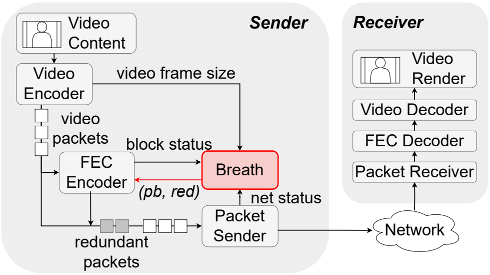

Figure 7: Workflow of Breath.

Workflow. As shown in Fig. 7, Breath works at the RTVS sender between the *Video Encoder* and the *FEC Encoder*. During transmission, video content is first encoded into frames by the video encoder and packetized into packets. The packets are then passed to *Packet Sender* and sent immediately, while copies of them are passed to FEC encoder for redundancy generation. After FEC encoding, the redundant packets generated are inserted into the sending queue. With the presence of Breath, when and how many redundant packets will be generated are controlled. After a new frame is encoded, Breath will take its frame size, frame interval and current network status, including RTT, sending rate and lost

[4] "Retransmission coupled with FEC" refers to a mechanism where retransmission is triggered only after the FEC recovery attempt fails.

rate as input and search for a best (pb, red) decision as illustrated in Alg. 1. The decision will then be passed to FEC encoder to control whether to finish the FEC block on the current frame and generate *red* redundant packets, or wait for the next frame.

3.4 System Implementation

We implement Breath in a RTVS system built on an open-source QUIC library [3]. The system operates as shown in Fig. 7 and incorporates the following components: (*i*) *FEC codec:* A high-performance Reed-Solomon (RS) [48] codec is adopted to represent the standard FEC codec. Note that Breath is agnostic to the FEC codec and can work with others (*e.g.,* XORFEC). (*ii*) *Video codec and adaptive bitrate (ABR):* A frame-level ABR mechanism adjusts the video encoding bitrate based on bandwidth estimates from the congestion controller. As video frame transmission latency is the main component of concern, we adopt a simulated video encoder to generate frames that meet the target encoding bitrates. (*iii*) *Loss recovery:* A fundamental retransmission recovery mechanism for the lost packets is deployed in the system, independent of the FEC scheme. (*iv*) *Network state estimation:* The system selects the maximum loss rate observed over 10 consecutive 100 ms measurement windows as the input loss rate. RTT is collected from transport-layer acknowledgments. The sending rate is obtained from the congestion controller Copa [5]. Crucially, the system paces all packets (both media and redundancy) to prevent self-inflicted traffic bursts.

Breath configuration. We set Breath's parameters in Eq. (1) and Eq. (2) as $\omega = 10$ and $\lambda = 2$. These values were empirically chosen to provide a balanced performance between visual quality, timeliness, and redundancy cost (see Appendix D). More configuration details are provided in Appendix B.

4 Evaluation

4.1 Setup

Trace-driven emulation testbed. We use Linux TC to replay network traces collected from our production system (Sec. 2), which contain available frame-level bandwidth and loss rate. Our testbed includes over 100 mobile video sessions, each lasting between 60 and 70 seconds, with over 20% lost frames. The video frame rate is set to 60 fps. To better evaluate the performance of FEC schemes, we configure the OWD to be 50 ms. This setting emulates a typical long-distance scenario, where high latency makes timely recovery more challenging and magnifies the differences between FEC strategies.

Metrics. We evaluate Breath using two categories of metrics. First, user QoE metrics: (*i*) *Average Video Bitrate:* The bitrate of media data, excluding redundancy. Higher values indicate better visual quality (validated by perceptual visual quality analysis in Appendix F). (*ii*) *Deadline-Missing Rate (DMR):* The fraction of frames whose transmission latency exceeds the interactivity deadline (150 ms in our setting), which is introduced in [31]. DMR captures tail latency and directly impacts user experience. (*iii*) *Video frame latencies (VFL):* The transmission latency of video frames. The average, 50th, and 95th percentiles of VFL values are reported. Second, FEC performance metrics: (*i*) *Redundancy Rate:* The ratio of redundant packets to media packets. (*ii*) *Recovery Rate:* The ratio of lost frames that are successfully recovered by FEC. (*iii*) *Recovery Latency:* The average transmission latency of lost frames recovered by FEC.

Baselines. We deploy the following baselines in our system (Sec. 3.4) and compare Breath against them:

- *WebRTC* [46]: The default FEC mechanism in the widely-used WebRTC framework [47]. We implement its redundancy decision policy and set the encoding block size to a single frame.
- *Tooth* [4]: Recent effort which represents the state-of-the-art performance among existing learning-based FEC solutions.
- *Mpb-WebRTC (Max Protection Boundary WebRTC)*: A variant of WebRTC that uses the same redundancy policy but always expands the FEC coding block to the maximum size allowed by the deadline, as described in Sec. 2.3.

4.2 Overall QoE Performance

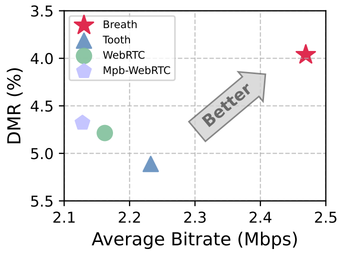

Figure 8: Trade-off between video bitrate (visual quality) and Deadline-Missing Rate (timeliness). Breath operates in the desirable top-right quadrant, achieving the best of both.

Breath significantly improves user QoE. Our primary finding is that Breath breaks the fundamental trade-off between visual quality (Bitrate) and interaction timeliness (DMR), establishing a new Pareto frontier. As illustrated in Fig. 8, Breath is the only scheme that operates in the optimal quadrant of highest bitrate and lowest DMR, achieving 2.47 Mbps and 3.96% respectively.

Superiority over single-frame FEC. Compared to single-frame schemes (WebRTC and Tooth), Breath leverages cross-frame protection to handle burst losses more efficiently. This translates into substantial QoE gains: Breath achieves a 14.4% higher bitrate than WebRTC (2.47 vs. 2.16 Mbps) and a 10.8% higher bitrate than Tooth (2.47 vs. 2.23 Mbps), while simultaneously slashing their DMRs by 17.3% and 22.5%, respectively.

Superiority over naive cross-frame FEC. Mpb-WebRTC, the strawman cross-frame approach, proves inefficient. Despite maximizing protection power, it delivers only a marginal DMR improvement over WebRTC (4.7% vs. 4.8%) and consumes the most redundancy (39.8%, see Sec. 4.3). In stark contrast, Breath achieves a 15.4% lower DMR and a 16.0% higher bitrate than Mpb-WebRTC. This highlights that simply maximizing the protection boundary is insufficient; an intelligent, fine-grained adaptation is necessary to achieve tangible QoE gains.

4.3 FEC Performance Breakdown

Here, we dissect the detailed video frame transmission latency distribution and underlying FEC metrics to reveal the source of Breath's QoE improvements. We show how its adaptive protection boundary masters the two fundamental trade-offs in FEC design: cost-efficiency and timeliness.

Cost-efficiency: adaptive redundancy. Fig. 9 reveals the source of Breath 's efficiency. Compared with single-frame FEC schemes, Breath achieves a higher recovery rate (91.8%) at a considerably

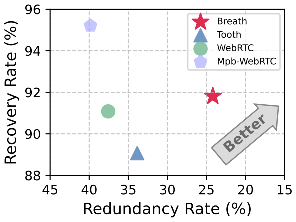

Figure 9: FEC protection cost-efficiency trade-off.

Alg.	Avg. (ms)	P50 (ms)	P95 (ms)
WebRTC	85	74	145
Tooth	86	75	153
Mpb-WebRTC	93	81	148
Breath (Ours)	**87**	**76**	**135**

Table 2: Transmission latency summary (smaller is better).

lower redundancy cost (24.2%). This is because Breath's adaptive boundary allows frames within a block to share their redundancy budget. A frame hit by a severe burst loss can be rescued by unused redundancy from its neighbors—redundancy that would otherwise be wasted in single-frame schemes. This enables Breath to handle severe loss events that would overwhelm single-frame schemes, achieving better protection with less overhead.

In contrast, while Mpb-WebRTC achieves the highest recovery rate (95.2%), it fails to translate this into a low DMR (Fig. 8). Its pitfalls are twofold. (*i*) High-latency, futile recoveries. By always using the maximum boundary, Mpb-WebRTC pushes recovery latencies dangerously close to the deadline. This leaves no margin for network jitter, causing many recoveries to complete after the 150 ms deadline has passed, rendering them useless for QoE. (*ii*) Inefficient, over-provisioned redundancy. Mpb-WebRTC fails to adjust its redundancy rate when expanding its protection boundary. It overlooks the cost-saving potential of cross-frame protection and, like WebRTC, over-provisions redundancy, leading to wasteful overhead. This highlights that a high recovery rate is meaningless if recoveries are not timely, a point we analyze next.

Timeliness: accelerating recovery. Tab. 2 shows that Breath strikes a superior latency balance. It achieves a significant 10 ms reduction in tail latency (P95) compared to the best baseline (WebRTC) at the cost of a negligible 2 ms increase in median latency (P50). This small increase in median latency is the price for unlocking the powerful benefits of cross-frame protection, a trade-off that proves highly effective. In a word, Breath's adaptive protection boundary advances the Pareto frontier between redundancy efficiency and low-latency recovery.

Conversely, Mpb-WebRTC suffers a significant degradation in median latency (7 ms worse than WebRTC) while barely improving tail latency. Additionally, we report the recovery latency of all schemes, where Mpb-WebRTC reaches a stunning 114 ms compared to 76 ms of single-frame schemes and 94 ms of Breath. As analyzed in Sec. 2.3, its largest protection boundary excessively delays all recoveries, overlooking the diminishing marginal benefit of increasing the block size.

5 Discussion and Future Work

Co-decision of target bitrate and redundancy rate. Following prior designs, Breath determines the redundancy rate after video frame encoding. This sequential decision may lead to slight bitrate fluctuations across consecutive frames. For future exploration, Breath can be improved by jointly determining the video bitrate and redundancy rate, such as combining the core idea of existing studies on co-decision mechanisms [2, 7].

Integration with visual quality evaluation. For computational efficiency, Breath adopts video bitrate as the visual quality optimization goal. However, Breath could also integrate with visual quality evaluation methods by replacing $\sum_{i=1}^{N} d(i)$ from Eq. (3) with visual quality metrics (*e.g.,* PSNR [17], SSIM [45], VMAF [35], etc.), providing a closer approximation of user's QoE.

Optimization of loss rate estimator. Breath achieves QoE improvements based on a simple loss rate estimator (the same as WebRTC). However, using an advanced estimator that statistically utilizes temporal correlation or captures loss characteristics based on deep-learning methods is also possible to further enhance Breath's adaptability and robustness in highly dynamic networks.

6 Related work

Previous studies have focused on accelerating retransmissions [9, 26, 34, 39], but relying solely on retransmission is often inadequate for meeting the stringent latency requirements of RTVS. Therefore, extensive works [2, 4, 6–8, 23, 31, 32, 36, 38, 42] have shifted their focus to optimizing FEC strategies for RTVS. Recent FEC schemes [6, 8, 23] employ deep neural networks to better adapt to network dynamics. Subsequent efforts [4, 38] characterize fine-grained packet loss patterns and optimize FEC redundancy allocation accordingly. Specifically, Tambur [38] reschedules FEC packets of one frame to later frames to improve recovery robustness under burst loss, but derives its constant interleaving interval (τ) from the frame deadline. This pushes recovery latency close to the deadline limit, sacrificing latency to ensure recovery probability (similar to Mpb-WebRTC in Sec. 2.3). Tooth [4] adjusts redundancy levels based on frame-level loss rate distribution according to frame sizes. Beyond solely optimizing FEC, there are also studies [2, 7, 31] that optimize FEC with other modules, such as retransmission, ABR, and congestion control. However, as illustrated in Sec. 2.2, existing solutions are fundamentally limited by inaccurate estimation of the frame-level loss rate, and thus cannot precisely allocate redundancy.

A related but distinct research field is that of error concealment techniques [19, 21, 22, 44, 55]. The goal of these studies is to mitigate the perceptual impact of loss without relying on redundancy or network-based retransmissions. Emerging approaches [10, 24] explore encoder-side resilience by redesigning neural video codecs to support loss-resilient frame delivery. These approaches can be applied with Breath to further improve the user's QoE.

7 Conclusion

This paper presented the first-of-its-kind adaptive protection boundary mechanism Breath for FEC encoding in RTVS. Breath dynamically adjusts the protection boundary and redundancy rate by minimizing the FEC protection overhead based on network and video dynamics. Implemented in a RTVS system and evaluated using traces collected from the production system, Breath significantly improves user QoE, pushing forward the Pareto frontier of loss recovery efficiency and overhead.

Acknowledgment

We thank all the reviewers for their insightful comments. This work is supported by National Key R&D Program of China (Grant No. 2022YFB2901800) and the Postdoctoral Fellowship Program of China Postdoctoral Science Foundation (Grant No. GZC20251071).

References

[1] 2023. *Ericsson Mobility Report, November 2023.* Technical Report. Ericsson. https://www.ericsson.com/4adb7e/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-november-2023.pdf Accessed: 2025-10-07.

[2] Ahmad Alhilal, Tristan Braud, Bo Han, and Pan Hui. 2022. Nebula: Reliable Low-Latency Video Transmission for Mobile Cloud Gaming. In *Proc. ACM Web Conf. 2022*. 3407–3417.

[3] Alibaba Inc. 2023. XQUIC: QUIC and HTTP/3 Implementation in C. https://github.com/alibaba/xquic. [Online; accessed 31-July-2025].

[4] Congkai An, Huanhuan Zhang, Shibo Wang, Jingyang Kang, Anfu Zhou, Liang Liu, Huadong Ma, Zili Meng, Delei Ma, Yusheng Dong, et al. 2025. Tooth: Toward Optimal Balance of Video QoE and Redundancy Cost by Fine-Grained FEC in Cloud Gaming Streaming. In *Proc. 22nd USENIX Symp. on Networked Systems Design and Implementation (NSDI 25)*. 635–651.

[5] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *Proc. 15th USENIX Symp. on Networked Systems Design and Implementation (NSDI 18)*. 329–342.

[6] Ke Chen, Han Wang, Shuwen Fang, Xiaotian Li, Minghao Ye, and H Jonathan Chao. 2022. RL-AFEC: Adaptive Forward Error Correction for Real-Time Video Communication Based on Reinforcement Learning. In *Proc. 13th ACM Multimedia Systems Conf.* 96–108.

[7] Sheng Cheng, Han Hu, and Xinggong Zhang. 2023. ABRF: Adaptive BitRate-FEC Joint Control for Real-Time Video Streaming. *IEEE Transactions on Circuits and Systems for Video Technology* 33, 9 (2023), 5212–5226.

[8] Sheng Cheng, Han Hu, Xinggong Zhang, and Zongming Guo. 2020. DeepRS: Deep-Learning Based Network-Adaptive FEC for Real-Time Video Communications. In *2020 IEEE Int'l Symp. on Circuits and Systems (ISCAS)*. 1–5.

[9] Sheng Cheng, Han Hu, Xinggong Zhang, and Zongming Guo. 2023. Rebuffering but Not Suffering: Exploring Continuous-Time Quantitative QoE by User's Exiting Behaviors. In *IEEE INFOCOM 2023 - IEEE Conf. on Computer Communications*. 1–10.

[10] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. GRACE: Loss-Resilient Real-Time Video Through Neural Codecs. In *Proc. 21st USENIX Symp. on Networked Systems Design and Implementation (NSDI 24)*. 509–531.

[11] Yihua Cheng, Ziyi Zhang, Hanchen Li, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, Mohammad Alizadeh, and Dina Katabi. 2023. AFR: A Flexible, Efficient, and General FEC Framework for Real-Time Video Communication. In *Proc. 20th USENIX Symp. on Networked Systems Design and Implementation (NSDI 23)*. 973–992.

[12] Cisco Public. 2020. *Cisco Annual Internet Report (2018–2023) White Paper.* White Paper. Cisco Systems. Accessed: 2025-10-07.

[13] Sadjad Faghani, Francis Y Yan, Ganesh Ananthanarayanan, Philip Levis, Carl Wiseman, Jeffrey C Mogul, Ali Ghodsi, Scott Shenker, and Hari Balakrishnan. 2018. Salsify: Low-Latency Network Video Through Stronger Just-in-Time Congestion Control. In *Proc. 15th USENIX Symp. on Networked Systems Design and Implementation (NSDI 18)*. 47–60.

[14] William Feller. 1991. *An Introduction to Probability Theory and Its Applications, Volume 2*. Vol. 2. John Wiley & Sons.

[15] Utkarsh Goel, Moritz Steiner, Mike P. Wittie, Martin Flack, and Stephen Ludin. 2016. HTTP/2 Performance in Cellular Networks: Poster. In *Proc. 22nd Annual Int'l Conf. on Mobile Computing and Networking*. 433–434.

[16] Xiangjie Huang, Jiayang Xu, Haiping Wang, Hebin Yu, Sandesh Dhawaskar Sathyanarayana, Shu Shi, and Zili Meng. 2025. ACE: Sending Burstiness Control for High-Quality Real-Time Communication. In *Proc. ACM SIGCOMM 2025 Conf.* 1182–1198.

[17] Quan Huynh-Thu and Mohammed Ghanbari. 2008. Scope of Validity of PSNR in Image/Video Quality Assessment. *Electronics Letters* 44, 13 (2008), 800–801.

[18] International Telecommunication Union. 2001. *ITU-T Recommendation G.1010: End-User Multimedia QoS Categories.* Recommendation. ITU.

[19] Hubert Jay, Joris De Vlaminck, and Olivier Bonaventure. 2020. NEL: A New Logic to Escape Packet Loss and Reordering in Interactive Networked Applications. In *Proc. ACM SIGCOMM Conf.* 427–441.

[20] Muhammad Asif Khan, Emna Baccour, Zina Chkirbene, Aiman Erbad, Ridha Hamila, Mounir Hamdi, and Moncef Gabbouj. 2022. A Survey on Mobile Edge Computing for Video Streaming: Opportunities and Challenges. *IEEE Access* 10 (2022), 120514–120550.

[21] Vineeth Shetty Kolkeri. 2009. *Error Concealment Techniques in H.264/AVC, for Video Transmission Over Wireless Networks.* Master's thesis. The University of Texas at Arlington.

[22] Abhishek Kumar, Francis Yaghmour Yan, Anirudh Sivaraman, and KV Rashmi. 2022. Swift: Delay is Provably-Low with a Simple Frantic Schedule. In *Proc. ACM SIGCOMM 2022 Conf.* 207–223.

[23] Insoo Lee, Seyeon Kim, Sandesh Sathyanarayana, Kyungmin Bin, Song Chong, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2022. R-FEC: RL-Based FEC Adjustment for Better QoE in WebRTC. In *Proc. 30th ACM Int'l Conf. on Multimedia.* 2948–2956.

[24] Tianhong Li, Vibhaalakshmi Sivaraman, Pantea Karimi, Lijie Fan, Mohammad Alizadeh, and Dina Katabi. 2023. Reparo: Loss-Resilient Generative Codec for Video Conferencing. *arXiv preprint arXiv:2305.14135* (2023).

[25] Xiaofei Liao, Hai Jin, Yunhao Liu, Lionel M Ni, and Dafu Deng. 2006. Anysee: Peer-to-Peer Live Streaming. In *Proc. IEEE INFOCOM 2006. 25th IEEE Int'l Conf. on Computer Communications.* 1–10.

[26] Dmitri Loguinov and Hayder Radha. 2001. On Retransmission Schemes for Real-Time Streaming in the Internet. In *Proc. IEEE INFOCOM 2001*, Vol. 3. 1310–1319.

[27] Zhicong Lu, Haijun Xia, Seongkook Heo, and Daniel Wigdor. 2018. You Watch, You Give, and You Engage: A Study of Live Streaming Practices in China. In *Proc. 2018 CHI Conf. on Human Factors in Computing Systems.* 1–13.

[28] David JC MacKay. 2005. Fountain Codes. *IEE Proc.-Communications* 152, 6 (2005), 1062–1068.

[29] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Proc. 21st ACM Internet Measurement Conf.* 229–244.

[30] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. 2022. Achieving Consistent Low Latency for Wireless Real-Time Communications With the Shortest Control Loop. In *Proc. ACM SIGCOMM 2022 Conf.* 193–206.

[31] Zili Meng, Xiao Kong, Jing Chen, Bo ocean Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking Packet Loss Recovery in Edge-Based Interactive Video Streaming. In *Proc. 21st USENIX Symp. on Networked Systems Design and Implementation (NSDI 24).* 907–926.

[32] Zili Meng, Chen Sun, Bo Wang, Yaning Guo, Min Zhang, and Mingwei Xu. 2021. Light-FEC: A Lightweight, Flexible, and Efficient FEC Framework for Real-Time Communication. In *Proc. 2021 ACM SIGCOMM 2021 Conf.* 646–659.

[33] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. 2020. UVG Dataset: 50/120fps 4K Sequences for Video Codec Analysis and Development. *Proc. 11th ACM Multimedia Systems Conf.* (2020).

[34] Sassan Pejhan, Mischa Schwartz, and Dimitris Anastassiou. 1996. Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media. *IEEE/ACM Transactions on Networking* 4, 3 (1996), 413–427.

[35] Reza Rassool. 2017. VMAF Reproducibility: Validating a Perceptual Practical Video Quality Metric. In *2017 IEEE Int'l Symp. on Broadband Multimedia Systems and Broadcasting (BMSB).* 1–2.

[36] Dan Rubenstein, Jim Kurose, and Don Towsley. 1998. Real-Time Reliable Multicast Using Proactive Forward Error Correction. In *Proc. NOSSDAV '98.*

[37] Michael Rudow, Francis Y Yan, Ganesh Ananthanarayanan, Abhishek Kumar, and K V Rashmi. 2023. Converge: A Versatile and Efficient Method for Streaming Neural Network Inferences. In *Proc. 20th USENIX Symp. on Networked Systems Design and Implementation (NSDI 23).* 1113–1130.

[38] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient Loss Recovery for Videoconferencing via Streaming Codes. In *Proc. 20th USENIX Symp. on Networked Systems Design and Implementation (NSDI 23).* 953–971.

[39] Pasi Sarolahti, Markku Kojo, and Kimmo Raatikainen. 2003. F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts. *ACM SIGCOMM Computer Communication Review* 33, 2 (2003), 51–63.

[40] Ryan Shea, Jiangchuan Liu, Edith C-H Ngai, and Yong Cui. 2013. Cloud Gaming: Architecture and Performance. *IEEE Network* 27, 4 (2013), 16–21.

[41] Amin Shokrollahi. 2006. Raptor Codes. *IEEE Transactions on Information Theory* 52, 6 (2006), 2551–2567.

[42] Gagan Singh, Jan Moll, Ilya Grigorik, and Ali C Begen. 2014. Enabling Client-Tailored Real-Time Video Streaming With Scalable Forward Error Correction. In *Proc. 5th ACM Multimedia Systems Conf.* 11–22.

[43] Shibo Wang, Shusen Yang, Xiao Kong, Chenglei Wu, Longwei Jiang, Chenren Xu, Cong Zhao, Xuesong Yang, Jianjun Xiao, Xin Liu, et al. 2024. Pudica: Toward Near-Zero Queuing Delay in Congestion Control for Cloud Gaming. In *Proc. 21st USENIX Symp. on Networked Systems Design and Implementation (NSDI 24).* 113–129.

[44] Yi Wang, Xiaoqiang Guo, Feng Ye, Aidong Men, and Bo Yang. 2013. A Novel Temporal Error Concealment Framework in H.264/AVC. In *2013 IEEE Int'l Conf. on Multimedia and Expo (ICME).* 1–6.

[45] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[46] WebRTC Project. 2023. *WebRTC Homepage.* https://webrtc.org/ [Online]. Available: https://webrtc.org/.

[47] WebRTC Project. 2023. WebRTC in Chromium. https://chromium.googlesource.com/chromium/src/+/e8f60fd53d98c1b1f67397209066e1453f6957ab/. [Online].

[48] Stephen B Wicker and Vijay K Bhargava. 1999. *Reed-Solomon Codes and Their Applications.* John Wiley & Sons.

[49] Jiyan Wu, Chau Yuen, Ngai-Man Cheung, Junliang Chen, and Chang Wen Chen. 2015. Enabling Adaptive High-Frame-Rate Video Streaming in Mobile Cloud Gaming Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 12 (2015), 1988–2001.

[50] Jiaxing Zhang, Qinghua Wu, Gerui Lv, Wenji Du, Qingyue Tan, Wanghong Yang, Kai Lv, Yuankang Zhao, Yongmao Ren, Zhenyu Li, et al. 2025. Predictable Real-Time Video Latency Control With Frame-Level Collaboration. In *2025 IEEE Real-Time Systems Symp. (RTSS)*. 1–15.

[51] Ziyi Zhang, Yihua Cheng, Kuntai Du, Hanchen Li, and Dina Katabi. 2023. ExStream: A Cross-Layer Design for Streaming 360-Degree Videos Over Wireless Networks. In *Proc. 29th Annual Int'l Conf. on Mobile Computing and Networking*. 1–15.

[52] Yuankang Zhao, Qinghua Wu, Gerui Lv, Furong Yang, Jiuhai Zhang, Feng Peng, Yanmei Liu, Zhenyu Li, Ying Chen, Hongyu Guo, et al. 2024. JitBright: Towards Low-Latency Mobile Cloud Rendering through Jitter Buffer Optimization. In *Proc. 34th Workshop on Network and Operating System Support for Digital Audio and Video*. 36–42.

[53] Yuankang Zhao, Furong Yang, Gerui Lv, Qinghua Wu, Yanmei Liu, Jiuhai Zhang, Yutang Peng, Feng Peng, Hongyu Guo, Ying Chen, et al. 2025. MARC: Motion-Aware Rate Control for Mobile E-Commerce Cloud Rendering. In *2025 USENIX Annual Technical Conf. (USENIX ATC 25)*. 217–232.

[54] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to Coordinate Video Codec With Transport Protocol for Mobile Video Telephony. In *Proc. 25th Annual Int'l Conf. on Mobile Computing and Networking*. 1–16.

[55] Jie Zhou, Bo Yan, and Hamid Gharavi. 2010. Efficient Motion Vector Interpolation for Error Concealment of H.264/AVC. *IEEE Transactions on Broadcasting* 57, 1 (2010), 75–80.

A Ethics

This work does not raise any ethical issues. All collected data are authorized by users, desensitized, and include performance-related information only.

B Other Configuration of Breath

For simplicity and calculation efficiency of Breath, we set the expected retransmission round $\mathbb{E}[rtx_i]$ in Eq. (8) as 1[5]. We normalize $\sum_{i=1}^{N} \mathbb{E}_{L_i}$ in Eq. (1) by dividing it by $N \cdot RTT_{min}$. For $d(\hat{i+1})$ in Alg. 1, it is derived from the latest target video bitrate from the ABR controller of the system.

C Impact of Video Encoder Overshoot

In practice, when adopting different video encoding strategies or encountering highly dynamic motion scenes, video frames may be significantly larger than the target encoding size (known as "encoder overshoot" [13, 16, 53, 54]), and they cannot be transmitted within a frame interval. Thus, the transmission time of the protection block in Eq. (6) and Eq. (8) should be replaced by:

$$\max\left\{\frac{MTU}{sr_i}\cdot\left(\sum_{j=i}^{N} d(j)+red\right),\ (N-i)\cdot I+\frac{MTU}{sr_i}\cdot(d(N)+red)\right\}.$$

D Parameter Sensitivity

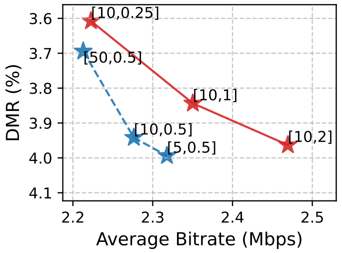

Figure 10: Impact of ω and λ on the trade-off between QoE metrics.

In Breath, ω governs protection boundary aggressiveness and λ regulates redundancy usage. In Fig. 10, the parameter sets along the red solid curve represent the Pareto-optimal frontier of Breath's performance. Among them, we select the parameter set [10, 2] to represent Breath for its superior spatial redundancy efficiency with a satisfactory latency performance.

(*i*) Timeliness vs. quality (λ). The parameter λ directly controls Breath's sensitivity to bandwidth overhead. A lower λ allows Breath to allocate more redundancy to enhance timeliness. As shown by the red solid curve in Fig. 10, lowering λ from 2.0 to 0.25 makes Breath prioritize timeliness over bandwidth efficiency. This improves DMR from 3.96% to 3.6%, at the cost of a 10% drop in average bitrate (from 2.47 to 2.22 Mbps).

(*ii*) Tail vs. median latency (ω). The parameter ω governs the aggressiveness of tail latency control. A higher ω value encourages Breath to adopt more aggressive protection boundary strategies to minimize deadline misses. As shown by the blue dashed curve in Fig. 10, increasing ω from 5 to 50 makes Breath more aggressive in controlling tail latency. This reduces DMR from 4.0% to 3.7% at the cost of a drop in bitrate and a slight increase in median latency.

For production deployment, Operators should increase ω when retransmission rates are high (indicating low FEC recovery rates), and decrease it when recovery latency is excessive. Increase λ to curb redundancy bandwidth consumption and decrease λ to utilize available bandwidth for higher reliability.

E Computational Overhead

All experiments were conducted on a Linux server equipped with an Intel Xeon Silver 4214 CPU @ 2.20 GHz running Ubuntu 20.04.6 LTS, and Breath's per-frame decision-making time was recorded. Our analysis shows a negligible mean computation latency of 0.47 ms, with 94.6% of computations completed within 1 ms, and 98.8% completed within 2 ms. Within the processing time of Alg. 1, 23.4% is spent on Step 1 and 76.5% on Step 2, while Step 3 takes only 0.1%. The near-instantaneous decision-making confirms that Breath can operate even in extremely demanding scenarios like 120 fps streaming.

F Perceptual Visual Quality Analysis

We extended our evaluation using VMAF, the industry-standard perceptual metric that accurately reflects subjective quality. We performed a curve fitting analysis of the bitrate-to-VMAF mapping on the UVG dataset [33]. Specifically, we encoded all 16 videos in the dataset at various bitrates to derive the fitted VMAF scores corresponding to specific bitrates. Results confirm Breath's 10.6%–14.2% bitrate gains effectively translate into perceptible visual improvements, achieving a 3.5%–4.7% VMAF increase. These results serve as a robust and reproducible proxy for subjective QoE, effectively demonstrating user experience enhancement.

[5] We observed in evaluations that different values of $\mathbb{E}[rtx_i]$ hardly affect Breath's performance advantages, whereas the computation overhead of solving for it is significant.