Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming

Gerui Lv[†§], Qinghua Wu[†§‡], Yanmei Liu, Zhenyu Li[†§‡], Qingyue Tan[†§], Furong Yang[†], Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, Gaogang Xie[§¶]
[†]Institute of Computing Technology, Chinese Academy of Sciences
[§]University of Chinese Academy of Sciences [‡]Purple Mountain Laboratories
[¶]Computer Network Information Center, Chinese Academy of Sciences

Abstract

Increasing bandwidth demands of mobile video streaming pose a challenge in optimizing the Quality of Experience (QoE) for better user engagement. Multipath transmission promises to extend network capacity by utilizing multiple wireless links simultaneously. Previous studies mainly tune the packet scheduler in multipath transmission, expecting higher QoE by accelerating transmission. However, since Adaptive BitRate (ABR) algorithms overlook the impact of multipath scheduling on throughput prediction, multipath adaptive streaming can even experience lower QoE than single-path. This paper proposes Chorus, a cross-layer framework that coordinates multipath scheduling with adaptive streaming to optimize QoE jointly. Chorus establishes two-way feedback control loops between the server and the client. Furthermore, Chorus introduces *Coarse-grained Decisions*, which assist appropriate bitrate selection by considering the scheduling decision in throughput prediction, and *Fine-grained Corrections*, which meet the predicted throughput by QoE-oriented multipath scheduling. Extensive emulation and real-world mobile Internet evaluations show that Chorus outperforms the state-of-the-art MPQUIC scheduler, improving average QoE by 23.5% and 65.7%, respectively.

CCS Concepts

• **Networks** → **Mobile networks**; **Cross-layer protocols**.

Keywords

Multipath QUIC, Adaptive Video Streaming, QoE

Co-first authors: Gerui Lv, Qinghua Wu. Corresponding authors: Zhenyu Li, Gaogang Xie. Email: {lvgerui, wuqinghua, zyli, tanqingyue22s, yangfurong}@ict.ac.cn, xie@cnic.cn.

ACM MobiCom '24, Nov. 18-22, 2024, Washington, D.C., USA

ACM ISBN 979-8-4007-0489-5/24/09.
https://doi.org/10.1145/3636534.3649359

ACM Reference Format:
Gerui Lv[†§], Qinghua Wu[†§‡], Yanmei Liu, Zhenyu Li[†§‡], Qingyue Tan[†§], Furong Yang[†], Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, Gaogang Xie[§¶]. 2024. Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming. In *Proceedings of The 30th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '24).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3636534.3649359

1 Introduction

Recent years have witnessed the popular trend of HTTP-based mobile video applications [71], from traditional video-on-demand (VoD) [35, 52, 95] and live video [38, 45, 78] to 360° panoramic video [18, 62, 84] and volumetric video [47, 81] that are applied in Augmented/Virtual/Mixed Reality (AR/VR/MR). To adapt the video quality to fluctuating bandwidth, HTTP-based adaptive streaming (HAS) is widely deployed in commercial video services [6, 7, 53]. HAS encodes video content into multiple quality (bitrate) representations, each divided into chunks of equal playback duration (usually 2-5 seconds). Adaptive BitRate (ABR) algorithms in the client player, which dynamically select the bitrate for each chunk based on throughput prediction, play a crucial role in optimizing the Quality of Experience (QoE), including maximizing video bitrate and minimizing rebuffering time.

The rapid increase in bandwidth requirements (tens [4, 68, 71, 84] to hundreds [29] of Mbps) of emerging video applications poses significant challenges to maintaining consistent high-quality delivery in mobile networks. As a promising solution, multipath transmission [25, 48] extends network capacity by aggregating the bandwidth of different wireless links (e.g., WiFi and 5G). Packet scheduler, the core component in multipath transmission, determines when, how, and in what order to assign packets to all subflows (referred to as "paths" in this paper), to optimize transport performance, i.e., higher throughput or shorter transmission time.

While multipath transmission is expected to provide performance no worse than the best single-path [65], our comprehensive evaluations in emulated and real-world mobile adaptive streaming (§5.2 and §5.5) show that even with higher chunk throughput, the ABR algorithm can still notably experience much lower QoE on multiple paths than

on a single path, especially in highly dynamic networks. This QoE degradation is attributed to frequent severe rebuffering events in multipath adaptive streaming, primarily due to incorrect bitrate selection rather than deficient transport performance. The root cause of this issue is that *ABR algorithms overlook the impact of multipath scheduling on perceived throughput, resulting in inaccurate throughput predictions.* Previous studies have widely reported the significant impact of multipath scheduling on application throughput [30, 43, 46, 70]. Yet, since scheduling typically operates after the bitrate selection for each video chunk, it is overlooked by ABR algorithms. Consequently, multipath scheduling introduces additional uncertainty into ABR throughput predictions, leading to incorrect bitrate decisions and ultimately catastrophic QoE degradation [57, 96].

To address this issue, our key idea is to *coordinate multipath scheduling and ABR algorithms to optimize QoE jointly.* This coordination aims to meet two necessary conditions for improving QoE in mobile adaptive streaming: *(i)* ensuring appropriate bitrate selection and *(ii)* providing transport performance that satisfies QoE requirements. While the idea may appear straightforward in principle, its practical implementation presents the following unique challenges.

(i) How can coordination between the server transport layer and the client application be achieved? The multipath packet scheduler and the ABR algorithm reside in distinct network protocol layers and endpoints. The former operates within the server protocol stack, whereas the latter is situated in the client video player. Although cross-layer coordination on a single side has been achieved in previous work [99], practically coordinating both protocol layers and endpoints remains challenging.

(ii) How can multipath scheduling enhance throughput prediction for ABR algorithms? Numerous studies on single-path adaptive streaming have emphasized that chunk throughput prediction is vital in ABR algorithms for QoE optimization [50, 56, 79, 89, 90, 95, 100]. However, according to our analysis, multipath scheduling complicates throughput prediction of ABR algorithms, leading to potential QoE degradation. A significant challenge arises because most multipath schedulers operate during each chunk's transmission. This prevents ABR algorithms from foreseeing scheduling decisions and incorporating them into throughput predictions. Regrettably, little effort has been directed to improve multipath throughput prediction in mobile adaptive streaming.

(iii) How can multipath scheduling satisfy the QoE requirements while minimizing costs? Multipath scheduling endeavors to strike a balance between transport performance and costs [30, 42, 98]. For instance, previous studies carefully control reinjection in multipath transmission (§2.1). However, without explicit awareness of QoE requirements, achieving this balance is difficult in multipath adaptive streaming. One underlying question is: To what extent of transport performance should be provided by multipath scheduling to avoid QoE degradation? If there is no clear guidance, even if the ABR algorithm accurately predicts throughput, multipath scheduling may struggle to meet the expected transport performance within cost limitations.

(iv) How can the coordination design be compatible with standard architectures with minimal effort? Both multipath transmission and mobile adaptive streaming have been already applied in large-scale production [6, 7, 10, 53, 98]. For massive deployment, it is important to ensure that this coordination design can be easily integrated into real-world systems. On the one hand, multipath scheduling should be able to interact with existing web servers with minimal modifications. On the other hand, the design should be accessible for application providers to adopt, without introducing prohibitive overhead, especially given the constrained computational capacity of mobile devices.

In this paper, we propose Chorus, a cross-layer framework that coordinates multipath scheduling with mobile adaptive streaming to optimize QoE jointly. Chorus introduces a novel design named **Coarse-grained Decisions and Fine-grained Corrections (CD&FC)** (§3) to ensure appropriate bitrate selection and provide expected-time-oriented transport performance. The CD phase (§3.3) achieves better multipath chunk throughput prediction by predetermining chunk-level packet scheduling decisions, and the FC phase (§3.4) meets the predicted throughput while balancing transport efficacy and cost considerations. To accomplish this, Chorus incorporates **two-way feedback control loops** between the server transport layer and the client application.

We have implemented Chorus using a user-space QUIC library [2] and integrated Chorus into a real-world mobile video system with minimal modifications (§4), encompassing both the web server [73] and the mobile video player [16].

Extensive evaluations in both emulated and wild mobile networks confirm the consistent superiority of Chorus in optimizing QoE, attributed to its better multipath throughput prediction and adequate transport performance with minimal costs (§5). In real-world scenarios, Chorus improves the average overall QoE by 65.7%∼114.4% over XLINK, a state-of-the-art QoE-driven multipath scheduler for short videos, and single path QUIC. Furthermore, Chorus relies on only a few assumptions and computational resources, making it robust and practical to deploy in mobile video applications.

In summary, this paper makes the following contributions: *(i)* Reporting the performance issue arising from adaptive streaming uncoordinated with multipath transmission; revealing the root cause and the fundamental solution (§2). *(ii)* Designing Chorus, a close-loop coordination framework that ensures effective bitrate control for mobile adaptive streaming (§3). *(iii)* Implementing Chorus based on multipath QUIC

and integrating it into a real-world mobile video system (§4). *(iv)* Thoroughly evaluating Chorus in mobile networks, confirming its consistently high performance (§5).

2 Background and Motivation

2.1 Background

ABR algorithms in video streaming. ABR algorithms are vital for the application layer to optimize QoE in mobile adaptive streaming [1, 35, 37, 52, 77, 89, 90, 95]. The simplest ABR algorithm (rate-based) selects a bitrate no greater than the predicted throughput [37]. State-of-the-art ABR algorithms, on the other hand, take both buffer occupancy and throughput predictions as input. They predict the perceived QoE of each decision using control-theory [76, 90, 95] or deep-learning [33, 51, 52] methods and select bitrates that maximize the overall session's predicted QoE. To achieve this, ABR algorithms must resolve the conflict between achieving high bitrate, low rebuffering time, and few bitrate switches.

Multipath transport mechanism. As the most widely used multipath technique of the transport layer, Multipath TCP (MPTCP) [25] is supported by commercial mobile devices, such as iOS [10] and Android [9] smartphones. However, MPTCP requires OS-level support, which presents challenges for mobile application providers to conduct customized optimization. As an alternative, Multipath QUIC (MPQUIC) [19, 48, 83, 98] takes advantage of the user-space property of QUIC transport protocol [36, 41], and thus can be easily modified and integrated into mobile applications.

The packet scheduler is the core component in multipath transmission that aims to optimize transport performance by determining when, how, and in what order to assign each packet to each path, usually at the RTT level. Both MPTCP and MPQUIC use MinRTT [66] as their default scheduler, which selects a path with the smallest RTT and available congestion window (CWND) to send packets each time. However, numerous studies have shown that MinRTT performs poorly on highly heterogeneous paths because it assigns excessive packets to slow paths (e.g., with higher RTT) and thus underutilizes fast paths. Previous works have proposed two ways to address this issue. *(i) Reinjection* [26, 30, 42, 66, 98]: By retransmitting packets of slow paths on fast paths, the scheduler can ensure multipath transport performance no worse than the best single path at the cost of redundancy. *(ii) Out-of-order sending for in-order arrival* [23, 30, 40, 46, 70, 72, 74, 87, 91]: By assigning more packets (depending on the prediction) to fast paths than their CWNDs, the scheduler can utilize the aggregated bandwidth of all paths, but this comes at the risk of prediction error.

2.2 Ineffective Multipath Adaptive Streaming

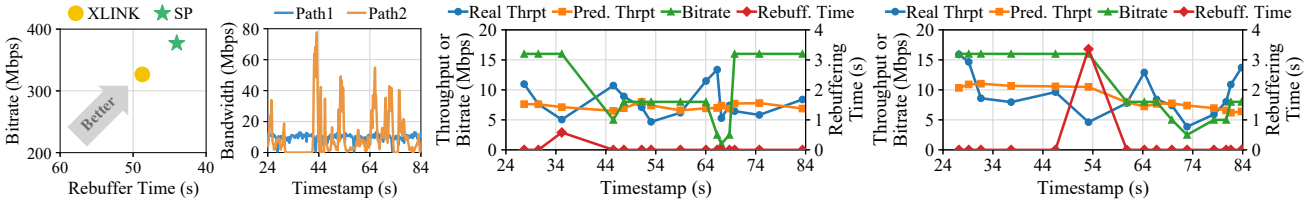
Multipath transmission seeks to optimize transport performance, i.e. higher throughput or shorter completion time, through packet scheduling [30, 46, 74, 98]. However, enhancing multipath scheduling independently does not necessarily induce QoE improvements for adaptive video streaming, which has its own control module (i.e., ABR algorithms) and optimization target (i.e., QoE metrics).

In our extensive testing on both emulated (§5.2) and real-world (§5.5) mobile networks, we evaluated multipath adaptive streaming using XLINK [98], a state-of-the-art packet scheduler. Notably, our results revealed that XLINK-based adaptive streaming underperforms compared to the best single-path (SP) in scenarios with highly fluctuating bandwidths. It showed an 11.7% lower bitrate and a 6.2% increase in rebuffering time on average, as depicted in Fig. 1a.

Higher throughput while lower QoE. The counter-intuitive results of multipath adaptive streaming can be explained by a case study. The test contains two paths over a 60-second span (24s-84s): Path one (the primary path) has bandwidths of 0.5-13.1 Mbps (avg. 10 Mbps, std. dev. 2 Mbps), and Path two ranges from 0-77.7 Mbps (avg. 10.4 Mbps, std. dev. 14.2 Mbps), as shown in Fig. 1b. This type of highly dynamic bandwidth is common in mobile networks, e.g., caused by handovers that occur between WiFi access points or cellular base stations. MPC [95] is used as the ABR algorithm and runs over SP and MinRTT+RI. MinRTT+RI allows unlimited packet reinjection (RI) when CWND is available, seen as *the upper-bound performance* for XLINK and MinRTT [98]. Fig. 1c and Fig. 1d show that MinRTT+RI indeed improves transport performance, achieving 17.5% higher average chunk throughput than SP over the 60 seconds. However, this improvement brings a 4.4% increase in total bitrates, but also a **4.7x** increase in total rebuffing time, eventually resulting in a 20.6% decrease in total QoE (Eq. 11).

Inaccurate multipath throughput prediction. Upon deeper analysis, we have identified that the reason accounting for these results is *multipath-based ABR algorithm is more likely to make incorrect decisions due to inaccurate throughput predictions*. Fig. 1d shows that MinRTT+RI-based ABR overestimates the real chunk throughput by **2.3x** at time 53. This prediction error leads to selecting the highest bitrate chunk (16 Mbps) and a severe rebuffering event lasting 3.4 seconds. In comparison, after a 0.6-second rebuffering event at time 35 due to a 41.2% overestimation, SP-based ABR quickly switches to lower bitrates for subsequent chunks and successfully avoids future rebuffering. By time 70, with a sufficient buffer accumulated, SP-based ABR resumes requesting the highest bitrate (Fig. 1c). MinRTT+RI's ABR algorithm shows more prediction errors, averaging 30.3%, compared to SP.

This phenomenon is pervasive in our tested scenarios, where total average path bandwidths fall below the top video bitrate, although MinRTT+RI achieves 3.9% higher bitrate than SP, it also increases the rebuffering time by 7.3%, resulting in reduced QoE (Fig. 10b). Additional results indicate that



(a) QoE Metrics (b) Paths' Bandwidth (c) A Case of SP-based ABR (d) A Case of MinRTT+RI-based ABR

Figure 1: Comparisons of multipath and single-path adaptive streaming performance in mobile networks.

MinRTT+RI is more prone to overestimating throughput than SP, evidenced by its 18.2% greater overestimation error rate at the 95th percentile (details omitted due to limited space).

2.3 Culprit: Overlooked Multipath Scheduling

Since ABR adopts the same throughput predictor for both MinRTT+RI and SP, each should have equivalent chances of prediction errors due to dynamic path bandwidth. Therefore, the observed *consistent overestimation* in multipath transmission mainly stems from its unique component: packet scheduling, which is analyzed by the following formulation.

When transmitting chunk k of size S_k over N paths, the average bandwidth of the n-th path is denoted as $B_k^{(n)}$. The chunk transmission time D_k is derived from the longest transmission times among all paths. Given that adaptive video streaming uses a video chunk as the basic transmission unit, all packets of an entire chunk must be received as a complete HTTP response on the client side before being passed to the video player. Hence, the chunk performance is primarily determined by the assignment ratio of packets over different paths (denoted as $\alpha_k^{(n)} \in [0,1]$, satisfying $\sum_{n=1}^{N} \alpha_k^{(n)} = 1$), rather than the packet transmission order. Consequently, the following equation holds for various packet schedulers:

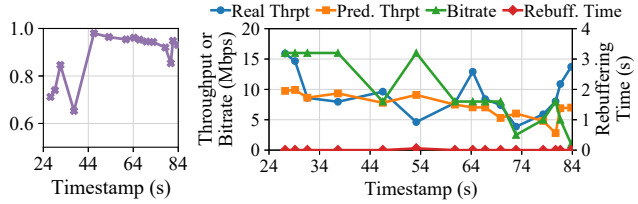
$$D_k = \max\{\alpha_k^{(1)} S_k / B_k^{(1)}, \ \ldots, \ \alpha_k^{(N)} S_k / B_k^{(N)}\} \ . \tag{1}$$

The chunk's throughput C_k is S_k/D_k [27, 50], leading to:

$$C_k = \min\{B_k^{(1)}/\alpha_k^{(1)}, \ \ldots, \ B_k^{(N)}/\alpha_k^{(N)}\} \ . \tag{2}$$

Eq. 2 clearly illustrates the impact of the scheduling decision on chunk throughput. Notably, even if the ABR algorithm precisely knows the bandwidth of all paths ($B_k^{(n)}$), it still cannot accurately predict chunk throughput without information on the packet assignment across paths, i.e., $\alpha_k^{(n)}$. Particularly, over-assigning packets to paths with lower bandwidth will result in diminished chunk throughput, causing the ABR algorithm to potentially overestimate actual throughput.

In situations of fluctuating link bandwidth, the assignment ratio of packets also varies for consecutive chunks. This is because the packet scheduler frequently adjusts scheduling decisions in response to the dynamic network environment. As presented in Fig. 2a, MinRTT+RI's assignment ratio on the primary path (path 1) for each chunk undergoes significant changes in the 60-second case. Specifically, at time 47, the ratio suddenly increases from 0.65 to 0.98, attributed to a



(a) Path Ratio (b) MinRTT+RI with Path Assignment Ratio

Figure 2: The Case of MinRTT+RI-based ABR performing with awareness of path assignment ratio.

sharp decline in the bandwidth of path 2 (Fig. 1b). However, the ABR algorithm overlooks this change, persistently overestimating the real throughput, and eventually triggering severe rebuffering at time 53 (the next chunk).

Multipath packet schedulers determine their decisions for each chunk after the bitrate selection and thus are overlooked by ABR algorithms. Indeed, *adaptive streaming operates as a prediction-based optimal control system [52, 95], whereas multipath scheduling is a black box to its core control module (ABR algorithms).* This increases prediction uncertainties and further results in control failures, i.e., inappropriate bitrate decisions that significantly impact sessions' QoE [57, 96].

2.4 Rescue: Incorporating Scheduling Information

The fundamental problem in multipath adaptive streaming is that: *Can the QoE be improved by considering scheduling information in throughput prediction?*

To answer this question, we introduced a new multipath throughput predictor for MinRTT+RI-based ABR, employing Eq. 2 with complete knowledge of multipath scheduling. The path bandwidth $B_k^{(n)}$ is predicted according to its receiving rate in past chunks (details in §3.3.2). While the actual packet assignment ratio can be obtained from offline MinRTT+RI logs, it is unattainable online. Through replaying the 60-second test in simulation (akin to [52]), results in Fig. 2b demonstrate that the ABR algorithm informed of multipath scheduling yields throughput predictions more aligned with reality. Particularly at the 47-second mark, sensing the scheduling change led to proactive throughput and bitrate adjustments. As a result, this new approach reduced rebuffering times to just 0.06s, enhancing the total QoE by 22.1% compared to the original MinRTT+RI-based ABR.

In summary, our analyses reveal that superior transport performance does not necessarily translate to enhanced QoE

for ABR algorithms. The root cause is that ABR algorithms are blind to the impact of multipath scheduling on perceived throughput, thereby leading to more prediction errors and further severe QoE degradation due to incorrect bitrate decisions. At a high level, adaptive streaming is uncoordinated with multipath scheduling in dynamic mobile networks. Although incorporating scheduling information into throughput prediction can improve QoE, applying this insight to the existing layered design is almost infeasible, prompting us to consider cross-layer coordination between multipath scheduling and ABR algorithms.

3 Chorus Design

We propose Chorus, a cross-layer framework that coordinates multipath scheduling with ABR algorithms to optimize QoE jointly. This section presents Chorus's design goals and challenges (§3.1), as well as an overview (§3.2) and detailed design of the framework (§3.3 and §3.4).

3.1 Design Goals and Challenges

Chorus takes two necessary conditions for ABR algorithms to optimize QoE as its goals: *(i)* ensuring appropriate bitrate selection and *(ii)* providing transport performance that meets the needs of ABR algorithms. However, two unique challenges in designing Chorus need to be addressed.

Challenge 1: *How can ABR algorithms be assisted in appropriate bitrate decisions under multipath scenarios?* While throughput prediction of ABR algorithms is vital for QoE optimization [50, 56, 79, 89, 90, 95, 100], multipath scheduling complicates chunk throughput prediction, as presented in §2.3. Notably, little work has been done to enhance multipath throughput prediction for adaptive streaming, leaving a gap in readily available solutions.

Solution: The transport mechanism determines the sending rate, eventually affecting the application's perceived throughput [13, 50]. As previous works have proved that mitigating this impact is feasible [82, 99], Chorus predetermines the scheduling decision for the entire chunk before transmission and informs ABR algorithms of this decision. This coordination reduces uncertainty in multipath throughput prediction. Chorus also considers bandwidth changes on each path and thus achieves better predictions, which directly assists ABR logic in making appropriate decisions.

Challenge 2: *How to provide transport performance that satisfies QoE requirements in mobile networks?* As ABR algorithms essentially make decisions that maximize predicted QoE [52, 95], they achieve optimal performance only if the transport layer guarantees the transmission time they expect. However, rapid changes in mobile networks during the chunk transmission may invalidate Chorus's previous one-shot chunk-level scheduling, resulting in increased transmission time and thus rebuffering events.

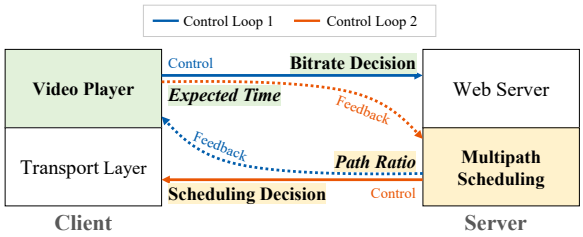


Figure 3: Chorus overview: two-way feedback control loops.

Solution: The difference in decision granularity between application (chunk-level) and transport layers (packet-level) provides an opportunity to correct the non-optimal decision by utilizing the fine-grained nature of the transport layer. If necessary, Chorus can reschedule and reinject packets during transmitting each chunk to accommodate ABR algorithms and address the previous decision error.

3.2 Chorus Overview

Chorus aims to optimize QoE for mobile multipath adaptive streaming by coordinating the server transport layer (multipath scheduling) with the client application (ABR algorithm).

To achieve this goal, Chorus incorporates **two-way feedback control loops**, as illustrated in Fig. 3. By utilizing the QOE_CONTROL_SIGNAL frame (QoE frame for short) in MPQUIC [49, 98], Chorus facilitates a bidirectional exchange of information between the server and client across different layers. As a result, both entities can collaboratively work towards a unified goal: optimizing QoE. Specifically, the client-side ABR algorithm predicts chunk throughput based on the server's pre-determined scheduling decision (*Path Ratio* in Fig. 3). In turn, the server packet scheduler adjusts its decision during chunk transmission in dynamic conditions, to meet the expected transmission time of the ABR algorithm (*Expected Time* in Fig. 3). Note that the two control loops operate on different time grains.

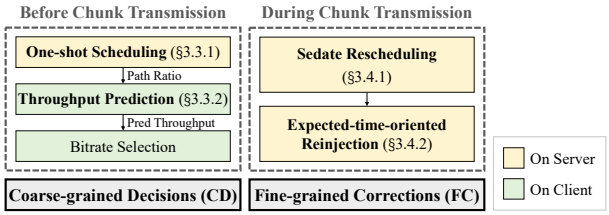


Figure 4: The submodules of Chorus's CD&FC.

Chorus introduces a novel design called **Coarse-grained Decisions and Fine-grained Corrections (CD&FC)**, which is depicted in Fig. 4. The CD phase takes place before transmitting a chunk, while the FC phase operates during the chunk transmission. In the CD phase (§3.3), Chorus first predetermines the scheduling decision of all packets in the chunk, then predicts throughput for this chunk based on that decision and selects the bitrate. Once the transmission of the

chunk starts, Chorus enters the FC phase (§3.4) and corrects the initial one-shot scheduling if it is non-optimal by *Sedate Rescheduling* and *Expected-time-oriented Reinjection*.

Chorus is illustrated with two paths for convenience, but not confined to any specific number of paths, which can be easily extended to more paths (see §6).

3.3 Coarse-grained Decisions (CD)

In the CD phase, Chorus coordinates the SERVER with the CLIENT to predetermine the chunk-level scheduling decision for bitrate selection. This process operates as follows: *(i) Informing Scheduling Decision*: Periodically (every 200ms, see §5.4), the SERVER sends a QoE frame to the CLIENT, encapsulating its latest scheduling decision, namely the path assignment ratio (α in Eq. 3). *(ii) Throughput Prediction*: Using the last received scheduling information, the CLIENT predicts the next chunk's multipath throughput, calculates the corresponding expected transmission time given the selected bitrate and sends it to the SERVER via a QoE frame. This expected time is utilized in the subsequent FC phase (§3.4). *(iii) Bitrate Selection*: The CLIENT feeds the ABR algorithm with the predicted throughput to select the next chunk's bitrate and makes an HTTP request for the corresponding chunk to the SERVER. *(iv) One-shot Scheduling*: Upon receiving the CLIENT's chunk request, the SERVER schedules packets of the entire HTTP response (chunk), assigning them across all paths at one time and then starting the chunk transmission.

Chorus's server asynchronously informs the client of the scheduling decision before it actually conducts the scheduling. While this design saves an RTT for the client to proactively inquire, it may introduce a bias between the client's received information and the server's actual scheduling. However, evaluations in §5.3 confirm that this bias is negligible.

3.3.1 *One-shot Packet Scheduling.*

In the CD phase, Chorus first conducts one-shot packet scheduling on the server side for the entire video chunk before transmission.

$$\alpha = \frac{B_f}{B_f + B_s} \quad . \tag{3}$$

Principle. The one-shot scheduling aims to achieve simultaneous completion on all paths, thereby minimizing the total transmission time for each video chunk. Given that the packet transmission order does not affect the chunk performance (§2.3), it is only important to determine the assignment ratio in Eq. 2. Specifically, Chorus assigns α of packets at the beginning and $1-\alpha$ of packets at the ending of a chunk to the fast and slow paths, respectively[1]. This assignment is performed only once per chunk and is not limited by CWND. Let B_f and B_s denote the average bandwidths of the fast and slow paths during chunk transmission, respectively. It is evident that C_k in Eq. 2 (as well as D_k in Eq. 1) can be optimized if and only if Eq. 3 is satisfied.

Practice. Chorus predicts B_f and B_s in Eq. 3 based on information from congestion control algorithms (CCAs) in the transport layer. For instance, BBR [13] maintains the predicted bandwidth of each path, while for Cubic [31] and other algorithms, the prediction can be calculated by dividing CWND by the smoothed RTT. The predicted bandwidth is further smoothed by EWMA to increase robustness. Additionally, to schedule packets of the entire chunk, Chorus is aware of the spatial and temporal boundary of each chunk (akin to [30]) using cross-layer APIs on the server side.

Note that Chorus does not expect this one-shot scheduling to be perfect during the transmission of the entire chunk due to possible network changes and invokes the subsequent FC phase (§3.4) to correct it if necessary.

3.3.2 *Throughput Prediction for ABR Algorithms.*

Based on the one-shot scheduling decision information, Chorus aims to provide better multipath throughput prediction for existing ABR algorithms on the client side.

Principle. Ideally, assuming that two paths complete the transmission simultaneously, the predicted chunk throughput $\hat{C}_k$ is directly derived from Eq. 2 where Eq. 3 holds, calculated as:

$$\hat{C}_k = \frac{\hat{B}_f}{\alpha} = \frac{\hat{B}_s}{1-\alpha}, \tag{4}$$

where $\hat{B}_f$ and $\hat{B}_s$ are predicted bandwidths of fast and slow paths, respectively, and α is given by the server's QoE frames.

Practice: A simple yet effective predictor. Although Eq. 4 is straightforward, its accuracy highly relies on whether its certain assumptions hold: *(i)* α should be optimal to complete the transmission on all paths simultaneously and *(ii)* $\hat{B}_f$ and $\hat{B}_s$ should be accurate. However, as illustrated in §3.3.1, Chorus does not guarantee assumption (i) holds under dynamic conditions. Additionally, past works have pointed out that the instant predicted bandwidth from the transport layer (e.g., the sending rate used in one-shot scheduling) tends to overestimate the link capacity and thus not suitable for directly used in the application decisions [13, 82, 90, 99], which can invalidate assumption (ii).

To make it practical for ABR algorithms, Chorus relaxes assumption (i) by using Eq. 2 to calculate predictions instead of Eq. 4. In addition, Chorus uses the local receiving rate (*RB*) of paths to predict their bandwidth rather than the server sending rate, thereby getting closer to assumption (ii). These factors are incorporated into the following equation:

$$\hat{C_k} = \min\{\frac{\hat{RB_f}}{\alpha}, \frac{\hat{RB_s}}{1-\alpha}\} \ , \tag{5}$$

where $\hat{RB_f}$ and $\hat{RB_s}$ are the receiving rate predictions of the fast and slow paths, respectively. *RB* of a path is calculated

[1] In Chorus, the fast path refers to the path with higher bandwidth, not lower RTT. Additionally, this assignment order is not necessary as long as the assignment ratio $\alpha/(1-\alpha)$ is met; it just facilitates implementation.

by dividing the bytes of each downloaded chunk received on that path by the chunk transmission time. Chorus uses the harmonic mean (HM) of the past five samples of RB to predict $\hat{RB}$, which is common in existing ABR algorithms [37, 95]. Although this prediction method[2] is simple, effective, and practical, we acknowledge that it is not a flawless solution and discuss potential enhancements in §6.

Compared with other prediction methods. To evaluate the performance of our proposed predictor (denoted as *Recv Rate*), we carried out a controlled emulation experiment (§5.1) and compared it to four other methods. *(i) HM*: the harmonic mean commonly used in single-path ABR algorithms [37, 95]; *(ii) RobustHM*: HM with error rate normalization proposed in RobustMPC [95] to provide conservative predictions; *(iii) Send Rate*: predicting throughput by Eq. 4 based on the latest sending rate (informed by the server); *(iv) Send Rate HM*, the HM-smoothed Send Rate. The results are shown in Fig. 5.

Fig. 5a shows that Recv Rate achieves the best average QoE performance, with 11.3%~58.5% improvements over others. In Fig. 5b, although Recv Rate is the most accurate, it only reduces 2.6% error rate compared to HM, while brings significant increase (26.1%) in QoE. To investigate this observation, we further introduce the *overestimation ratio*, which measures the chance of a predictor overestimating the actual chunk throughput. Fig. 5b shows that Recv Rate is more conservative, with a lower overestimation ratio than other predictors except for RobustHM.

To summarize, Recv Rate is accurate because it can detect and timely react to underlying network or scheduling changes on paths, evidenced by the case in Fig. 2b; it is also relatively conservative because it relaxes assumption (i) in Eq. 4. While previous works have been devoted to achieving absolutely accurate predictions in adaptive streaming [50, 56, 79, 89, 90, 100], these results bring a novel insight:

Insight 1: *Accurate but relatively conservative predictions assist ABR algorithms in optimizing QoE under dynamic conditions.*

Conservative predictions help avoid severe rebuffering events that significantly impact QoE [57, 96] under dynamic networks. However, the comparison with RobustHM indicates that being too conservative may miss the chance of increasing bitrates, and thus not maximize QoE, which suggests that the predictor should not sacrifice accuracy.

3.4 Fine-grained Corrections (FC)

During the transmission of each chunk, Chorus enters the FC phase to cope with the possibility that the one-shot decision made in the CD phase is non-optimal due to prediction error. In the FC phase, Chorus needs to provide adequate transport performance by limited but effective corrections.

To this end, Chorus conducts *two-stage corrections* in the FC phase, as illustrated in Fig. 6 and Alg. 1. The *1st stage* begins with the chunk transmission. In this stage, Chorus aims to fully utilize the bandwidth of all paths under dynamic conditions by the **Sedate Rescheduling** (§3.4.1, lines 2-9 in Alg. 1). When the transmission time exceeds a deadline base on ABR logic's expected time (lines 10-13 in Alg. 1), Chorus enters the *2nd stage*. During the last RTT of the chunk transmission (when all packets are sent), Chorus conducts the **Expected-time-oriented Reinjection** (§3.4.2, lines 14-18 in Alg. 1) for inflight packets to accelerate the transmission on the premise of minimizing traffic redundancy.

Our exhaustive evaluations in §5.3 show the designs of Fine-grained Corrections assist Chorus in optimizing QoE with infrequent rescheduling and limited reinjection.

3.4.1 1st stage: Sedate Rescheduling. The 1st-stage correction involves only Sedate Rescheduling, which is activated by the following event: when one path has extra CWND available to send more packets (i.e., no assigned but unsent packets left), while another path still has unsent packets beyond its current CWND (lines 2-5 in Alg. 1), indicating that the total bandwidth is underutilized. In this case, Chorus retrieves all unsent packets from the respective paths and reschedules them based on the latest bandwidth. Chorus recognizes that *frequently adjusting scheduling decisions is unnecessary when no CWND is available across all paths.* Therefore, Chorus refrains from immediate reactions to path-specific network changes during chunk transmission.

Chorus conducts Sedate Rescheduling in the same way as one-shot scheduling (§3.3.1), only with a difference in calculating the new assignment ratio. This difference arises from inflight packets on the path requiring one RTT to be acknowledged. Consequently, Chorus must consider the path's RTT in rescheduling and calculate the assignment ratio α by solving the following equation:

$$\frac{\alpha S_u}{B_f} + RTT_f = \frac{(1-\alpha)S_u}{B_s} + RTT_s \ , \tag{6}$$

where S_u indicates the size of unsent data of the chunk, RTT_f and RTT_s are RTT of the fast (with higher bandwidth, not shorter RTT) and slow paths, respectively. Note that the fast and slow paths are re-identified according to the latest bandwidth measure. Given $0 \le \alpha \le 1$, Eq. 6 is solved as:

$$\alpha = clip[\frac{B_f}{B_f + B_s} + \frac{B_f * B_s * (RTT_s - RTT_f)}{S_u * (B_f + B_s)}, 0, 1] \ , \tag{7}$$

where $clip[x, 0, 1]$ means limiting x to $[0, 1]$. In most cases, Sedate Rescheduling will not be triggered frequently, with only several or a dozen times in a chunk (§5.3). Note that if the RTT_f is much longer than RTT_s or S_u is too small, where Eq. 6 has $\alpha < 0$ or $\alpha > 1$, two paths may not complete

[2]In practice, when packets are barely assigned to the slow path and $\hat{RB}_s \approx 0$, $\hat{C_k}$ may be abnormally small. To avoid this issue, the predicted throughput can be further limited as $\hat{C_k} \ge \max\{\hat{RB}_f, \hat{RB}_s\}$, indicating Chorus can provide aggregated throughput no less than the best single path's bandwidth.

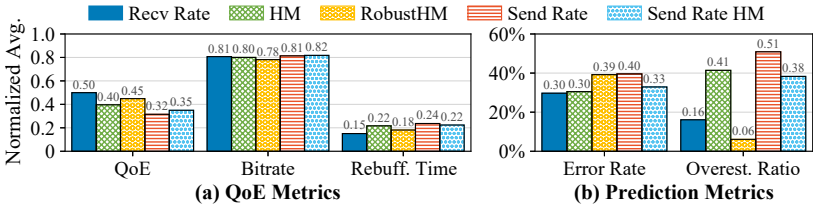


Figure 5: Performance comparison between the proposed predictor (Recv Rate) vs. others. QoE metrics are normalized by decimal scaling.

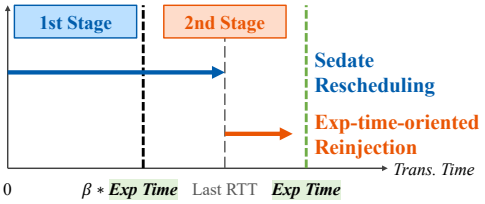


Figure 6: Timeline of Chorus during the Fine-grained Corrections (FC) phase.

Algorithm 1: Two-stage Corrections in FC

Input : *enable_renj*: if enabling reinjection; $\hat{D}_k$: player's predicted transmission time of chunk k; P_f: the fast path; P_s: the slow path; T_s: the time when server receives the request

begin
  // 1. Sedate Rescheduling in both the 1st and 2nd stages
  $extra\_cwnd_f \leftarrow P_f.cwnd - P_f.inflight - P_f.unsent$
  $extra\_cwnd_s \leftarrow P_s.cwnd - P_s.inflight - P_f.unsent$
  $enable\_resche \leftarrow$ XOR $(extra\_cwnd_f > 0 == True,$ $extra\_cwnd_s > 0 == True)$
  if *enable_resche* **then**
    $all\_unsent\_pkts \leftarrow$ retrieve_unsent_pkts (P_f, P_s)
    // calculate the ratio using Equation (7)
    $\alpha \leftarrow$ calc_resched_ratio $(all\_unsent\_pkts, P_f, P_s)$
    sched_pkts $(P_f, all\_unsent\_pkts, \alpha)$
    sched_pkts $(P_s, all\_unsent\_pkts, 1-\alpha)$
  // 2. Expected-time-oriented Reinjection in the 2nd stage
  if *enable_renj == False* **then**
    // calculate $\hat{E}_k$ using Equation (10)
    $exp\_time \leftarrow$ calc_exp_time $(\hat{D}_k, P_f, P_s)$
    if now () - T_s >= *exp_time* **then**
      enable_renj $\leftarrow$ *True*
  if *enable_renj == True* **then**
    // reinject inflight on another path if CWND allows
    if $!P_f.unsent$ **then**
      reinject $(P_s.inflight, P_f)$
    if $!P_s.unsent$ **then**
      reinject $(P_f.inflight, P_s)$

the transmission simultaneously. To cope with these cases, Chorus further conducts reinjection in the 2nd stage.

3.4.2 2nd stage: Expected-time-oriented Reinjection. The 2nd-stage correction involves Expected-time-oriented Reinjection to accelerate transmission in the last RTT, to meet the needs of ABR algorithms. Recalling §2.1, the state-of-the-art ABR logic can be formulated by the following equation [52, 95]:

$$R_k = \underset{r_m, 1 \le m \le M}{\arg\max} \sum_{k=1}^{K} \hat{QoE}(R_{k|r_m}), \tag{8}$$

where K denotes total number of chunks in a session, M denotes total number of bitrate levels, r_m denotes the bitrate of level m, and $R_{k|r_m}$ indicate selecting bitrate r_m for chunk k. Considering Eq. 11, the predicted QoE ($\hat{QoE}$) is as follows:

$$\hat{QoE}(R_k) = R_k - \max(\mu(\hat{D}_k - L_k), 0) - \lambda |R_k - R_{k-1}|, \tag{9}$$

where $k \ge 2$, $\hat{D}_k$ is the predicted transmission time, L_k is the buffer level when requesting chunk k, and the middle item indicates rebuffering time. Eq. 8 and Eq. 9 indicate that *ABR logic essentially relies on predicted transmission time*. Therefore, ABR algorithms need transmission time not to exceed their expectations as a necessary condition to optimize QoE. This observation further leads to the following insight:

Insight 2: *Controlling reinjection based on the expected time of ABR algorithms is an essential way to optimize QoE for adaptive streaming.*

Previous works control reinjection based on various factors, such as probability [66], specific threshold [30], confidence interval [42], or player's buffer level [98]. In contrast, Chorus conducts Expected-time-oriented Reinjection, which is effective and directly satisfies QoE requirements of ABR algorithms. Specifically, Chorus's server sets a deadline ($\hat{E}_k$) during the FC phase based on the expected time, calculated as:

$$\hat{E}_k = \beta * \hat{D}_k - (\alpha * RTT_f + (1-\alpha) * RTT_s) \ , \tag{10}$$

where β is an adjustment parameter that is set to 0.9 (see §5.4), and the last item indicates the average RTT between the client and the server. Recalling that $\hat{D}_k$ is given by the client via a QoE frame before requesting chunk k (§3.3).

During transmitting each chunk, Chorus records the transmission time on the server side and checks if it exceeds $\hat{E}_k$. If so, Chorus enters the 2nd stage and reinjects inflight packets in the last RTT of the transmission, by checking if all packets are sent and at least one path has available CWND.

4 Implementation and Deployment

Chorus is implemented with ~1000 lines of C code based on the multipath version of XQUIC [2], a user-space QUIC library. As the bottleneck in mobile networks typically occurs in the last mile [30, 42, 54, 98], multiple paths usually do not share the same bottleneck link. Therefore, Chorus uses decoupled CCA for each path, following [30, 32, 98]. Specifically, Chorus employs Cubic [31] (with slow-start-restart disabled [8, 52]), the default CCA of the Linux kernel. The choice of CCA is discussed in §5.4.

Emulation testbed. To evaluate Chorus in emulation, we build a testbed including a simple server application and a virtual client video player based on XQUIC [2]. Since our focus is solely on network conditions and the requesting behavior of video streaming in the emulation, we implement the ABR logic and the chunk request logic (as in dash.js [69])

but leave out the actual decoding and rendering process. We use Mahimahi [58] and its multipath version mpshell [55] to replay the traces collected in mobile networks (§5.1).

Deployment in the real world. We build a real-world platform to evaluate Chorus in the wild mobile Internet, including both the server and the mobile client player. Since Chorus is based on user space QUIC protocol, it is easy to integrate into applications. The server runs Tengine Web Server [73] with Chorus deployed (with only 49 lines modification of C Code), hosting video chunks. For the client, Chorus is integrated into MediaPlayer-Extended [16], an Android DASH [39] video player. MediaPlayer-Extended uses the OkHttp library (TCP-based) for HTTP requests, which is seamlessly replaced by TekiXquic [93], an XQUIC-based HTTP library. As per [32, 46], the WiFi link is selected as the primary path by default in Android. Finally, one 4G and two 5G Android smartphones are used to run the player APP.

5 Evaluation

This section thoroughly evaluates Chorus in both emulation and real-world mobile Internet.

5.1 Setup

Video sources. Big Buck Bunny [12] is used as the source video to evaluate various algorithms in both emulation and real-world mobile Internet. Following YouTube recommended encoding settings [68], the video is encoded at bitrates [1, 2.5, 5, 8, 16] Mbps with H.264/MPEG-4 codec by FFmpeg [24], corresponding to resolutions [360p, 480p, 720p, 1080p, 1440p (2K)]. At each bitrate level, the video is split into 4-second duration chunks by Bento4 [5]. Other videos, including Elephants Dream [21] and Sintel [75], are also evaluated in §5.4. The standard deviation of the highest bitrate chunk sizes of these three videos is 2756, 4276, and 3597 (KB), respectively.

Baselines. Since Chorus can be used with any ABR algorithms with throughput prediction, the representative one named MPC [95] is chosen, which uses harmonic mean (HM) as the predictor[3]. MPC is implemented in both the emulation testbed and the real-world video system (§4). Four QUIC-based transport schemes for MPC are considered as baselines, including three multipath schemes and one single-path scheme: *(i) MinRTT* [66]: is the basic scheduler in MPQUIC, which selects the path with the shortest RTT. *(ii) XLINK* [98]: is the state-of-the-art QoE-driven MPQUIC scheduler, which utilizes MinRTT in scheduling and controlling reinjection based on the player's buffer level. *(iii) MinRTT+RI*: represents the best transport performance of MinRTT and XLINK by conducting unlimited reinjection in the last RTT. *(iv) SP*: is the single path QUIC. We implemented XLINK using its original codes and set its double thresholds for the player's buffer level according to the recommendations in XLINK's paper, i.e., (95, 80) percentile of the buffer level distribution, which corresponds to (0.2s, 3.7s) in our case. Note that the only difference between MinRTT, XLINK, and MinRTT+RI is reinjection. Other mechanisms of XLINK are enabled as default for all these multipath baselines.

Performance metrics. Following [1, 34, 52, 95], we use the typical form in evaluating the QoE of video streaming, defined as the following equation:

$$QoE = \sum_{k=1}^{K} R_k - \mu \sum_{k=1}^{K} T_k - \lambda \sum_{k=1}^{K-1} |R_{k+1} - R_k| \ , \tag{11}$$

where the three items represent video quality, rebuffering time (T_k, in second), and smoothness of quality switch in a session, respectively. K is the total number of chunks in that session, and R_k is the bitrate (in Mbps) of chunk k. μ and λ are coefficients, set to the highest bitrate (16 in our case) and 1.0, respectively, following previous works [1, 34, 52]. In our emulation, the upper-bound session QoE is 1328, indicating 83 chunks transmitted at 16 Mbps with zero rebuffering.

Network traces. We collected 52 traces by saturatr [86] in mobile networks, including 47 cellular and 5 WiFi traces, with the average bandwidth matching the bitrate range of 1∼16 Mbps to ensure non-trivial bitrate selection [52]. Half of the traces were collected in stationary scenarios (e.g., outdoors, at home, or in the office), while others were collected during movement (e.g., walking, driving a car, or on high-speed rails). Each trace's uplink and downlink bandwidth were recorded, as well as the RTT and loss rate (via ping).

When replaying these traces in the emulation testbed (§4), we set the RTT and loss rate of a link (including uplink and downlink) according to its original records and set the buffer size of the link as 3×BDP. Other buffer settings are evaluated in §5.4. We randomly select two traces from the 52 available to create a combination of two paths, resulting in 26 tests, each containing a 5-minute video session. For SP, we take its higher QoE on the two paths as its performance. Fig. 7 depicts the individual and total downlink bandwidth of paths in each test. In 10 out of 26 tests, the total bandwidth of paths cannot support delivering video at the constant highest bitrate.

5.2 Trace-driven Evaluation

We begin by evaluating how Chorus performs in assisting the existing ABR logic compared to baselines in the emulation.

QoE performance. The QoE of Chorus and baseline schemes in the emulation are shown in Fig. 8a. Results demonstrate that Chorus successfully assists MPC in optimizing QoE. Compared to XLINK, MinRTT+RI, MinRTT and SP, Chorus improves the average QoE by 23.5%, 21.1%, 247.3% and 93%, respectively. In particular, Chorus achieves QoE improvement with respect to XLINK in 73.1% of all sessions (not shown due to space limitation). Regarding underlying

[3]Other predictors have also been evaluated (results omitted due to limited space). The sum of paths' receiving rate for multipath schemes yields similar QoE to HM, while RobustHM performs worse than HM for all baselines.

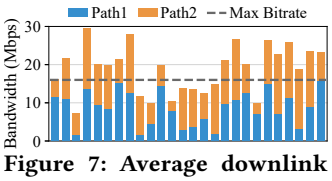
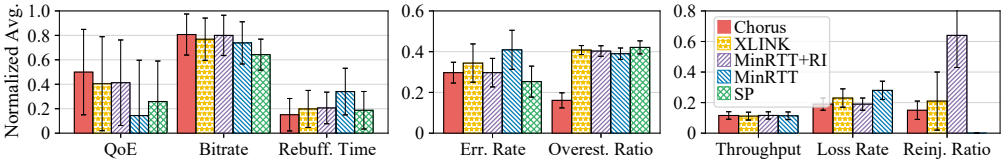


Figure 7: Average downlink bandwidth in each test.



(a) QoE Performance (b) Prediction Performance (c) Transport Performance

Figure 8: Overall performance of Chorus vs. baselines in emulation (with 95% confidence).

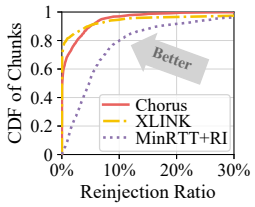
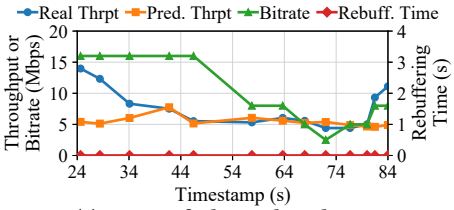
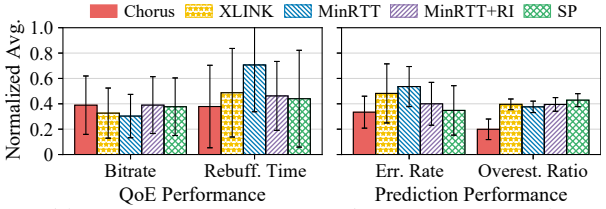


Figure 9: Reinjection ratio of each chunk.

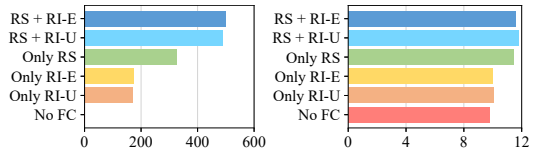


(a) A case of Chorus-based MPC



(b) QoE and prediction metrics (with 95% confidence)

Figure 10: Chorus vs. baselines under the weak scenario in emulation.



(a) QoE Sum (b) Throughput (Mbps)

Figure 11: Ablation study of Chorus's FC design.

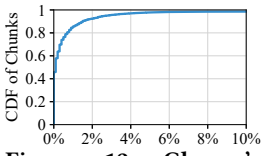


Figure 12: Chorus's asynchronous bias rate.

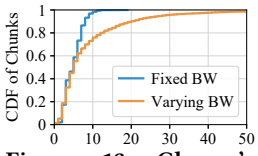


Figure 13: Chorus's rescheduling count.

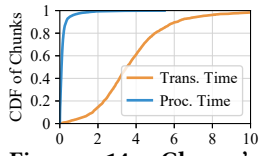


Figure 14: Chorus's chunk processing time.

QoE metrics, Chorus delivers 0.9%~9.27% higher bitrates than other multipath schemes and 25.6% more than SP on average. Meanwhile, Chorus reduces the average rebuffering time by 19.5%~55.7% compared to all baselines.

Prediction performance. Fig. 8b investigates Chorus's prediction performance. As shown, Chorus provides the most accurate predictions of all multipath schemes, with the lowest overestimation ratio. These results are consistent with the insight in designing Chorus (§3.3.2) and suggest that Chorus enhances multipath throughput predictions for the ABR algorithm, accounting for the lowest rebuffering time and highest overall QoE. This observation demonstrates that *Chorus has successfully realized its first goal of ensuring appropriate bitrate selection for ABR algorithms.*

Transport performance. Even though Chorus targets adequate rather than optimal transport performance, Fig. 8c shows that Chorus achieves the same transport performance as MinRTT+RI, which is the upper bound of XLINK and MinRTT's performance. Compared to XLINK, Chorus slightly improves throughput (3.7%) and effectively reduces the loss rate (17.4%) with fewer reinjection packets. These improvements are mainly attributed to Chorus's out-of-order sending for in-order arrival scheduling, showing that *Chorus has reached its second goal of satisfying the needs of ABR logic.*

Reinjection control. Chorus' innovative reinjection control and that of the state-of-the-art scheme XLINK are evaluated, as shown in Fig. 8c. Chorus reinjects 28.6% and 76.6% fewer packets while improving QoE over XLINK and MinRTT+RI, respectively. Fig. 9 further illustrates that Chorus significantly reduces the reinjection ratio at the tail, with a 34.1% and a 59.1% reduction at the 95th and 99th percentile relative to XLINK, respectively. These results suggest that *Chorus's expected-time-oriented scheme is more effective than XLINK's buffer-based scheme in adaptive streaming.*

Behavior under the weak scenario. Fig. 10a demonstrates the superiority of Chorus over baselines in the weak scenario[4], the same as that of the case study in Fig. 1. Specifically, Chorus predicts throughput quite accurately in timeslots 42-80, with only a 9.6% average error rate. Benefiting from this, Chorus successfully avoids rebuffering events and achieves the highest QoE with an 8.68% and 36.9% improvement over SP and MinRTT+RI, respectively. In fact, in the 10 tests where two paths' total average bandwidth is below the highest bitrate (Fig. 7), Chorus outperforms all baselines in bitrate, rebuffering time, and prediction error rate (Fig. 10b).

5.3 Chorus Deep Dive

In this part, controlled experiments are conducted to assess how individual designs of Chorus affect performance.

Ablation study of FC. We evaluate the impact of Chorus' FC phase including Sedate Rescheduling (RS) and Expected-time-oriented Reinjection (RI-E), using Unlimited reinjection (RI-U) for reference. Results in Fig. 11a show that *the complete FC design (RS + RI-E) is vital for Chorus to optimize QoE in mobile networks.* Specifically, disabling RS and RI-E reduces QoE by 65.2% and 34.4%, respectively. When FC is completely disabled (No FC), QoE nearly drops to 0 due to significant rebuffering. Similar trends appear in other metrics, such as chunk throughput (Fig. 11b), bitrate, rebuffering time, and

[4]XLINK achieves lower QoE than MinRTT+RI in this scenario, not shown.

error rate. In addition, RI-E achieves comparable QoE performance to RI-U (within 2% difference), with 59.5%~62.5% fewer reinjected packets, highlighting the efficiency of Chorus's reinjection control.

Asynchronous scheduling bias. In the CD phase, Chorus's server informs the client before its one-shot scheduling, potentially introducing bias due to network dynamics. Fig. 12 evaluates this, showing the relative change rate of the path ratio between the client-received information and the server's eventual scheduling decision. The results indicate that this bias is negligible, with a mean of 0.6% and a median of merely 0.1%. For 85.6% of the chunks, the bias is under 1%.

Rescheduling frequency. Chorus conducts rescheduling only when necessary. Fig. 13 shows that rescheduling is infrequent, even under varying bandwidths (our collected traces), with a mean and median value of 9.3 and 5, respectively. The count is less than 20 in 90.1% chunks. Additional results under fixed bandwidth (3 different bandwidth combinations and 3 different RTT combinations, generating 9 tests) suggest that rescheduling is mainly (about 60%) caused by inaccurate bandwidth predictions of the CCA (Cubic), which is further discussed in §5.4.

Path ratio in prediction. To investigate the effect of scheduling information (i.e., path ratio) on multipath throughput prediction, we evaluate Chorus using an alternative predictor that simply sums the receiving rates of the two paths, representing the aggregated bandwidth. The results (omitted) indicate that the path ratio information is necessary for achieving better throughput prediction. Specifically, Chorus's original predictor outperforms the alternative one by achieving 11.6% higher QoE, with similar bitrates (within 1% difference) and 19.8% lower rebuffering time.

Computational overhead. Chorus is a lightweight userapace framework, introducing negligible computational overhead when integrated into mobile applications. Tests on a Ubuntu 14.04 virtual machine show that Chorus server's average processing time for each chunk is 172ms, accounting for only 4.5% of the average chunk transmission time (3.9s), as presented in Fig. 14. Additionally, within the total processing time per chunk, approximately 40.8% is spent on packet transmission, including activities like interacting with the application and invoking the kernel UDP socket APIs.

5.4 Sensitivity Analysis

We in this part test the sensitivity of Chorus performance to different experimental environments and parameter settings.

Link buffer size. We utilized a 3xBDP link buffer in our main emulation evaluations (§5.2 and §5.3). Here, we further evaluated Chorus using a range of buffer sizes, spanning from shallow (1xBDP) to deep (10xBDP). The results in Fig. 15a show that the QoE performance of all schemes degrades under a 1xBDP buffer, but stabilizes as the buffer size exceeds 2xBDP. Notably, the buffer size does not impact the relative performance difference between these schemes. Specifically, even compared to XLINK's optimal performance (5xBDP), Chorus still achieves a 15.3% improvement in QoE. These results substantiate the consistent performance advantage of Chorus across different buffer sizes.

Congestion control algorithm (CCA). We observed that when paired with BBR, Chorus consistently performs worse than with Cubic (when the link buffer size is larger than 2xBDP), exhibiting a QoE degradation of 6.4% to 11.2%, as illustrated in Fig. 15. The root cause is that multipath scheduling confuses BBR's app_limited check logic [14], resulting in inaccurate link bandwidth prediction. In contrast, other MinRTT-based baselines rely solely on RTT, without incorporating BBR's prediction. As a result, their performance remains similar regardless of whether Cubic or BBR is used as the CCA when the link buffer size exceeds 2xBDP. While using alternative CCAs such as Copa [3] for Chorus may yield better performance, exploring these options falls beyond the scope of this paper.

Source video. We include two additional open-source videos used in previous studies [50, 63, 80] for comparison: Elephants Dreams (ED) [21] and Sintel [75]. These videos are encoded in the same way as BBB but have different chunk size distributions (§5.1). Fig. 16 shows that BBB and ED yield similar results, while all schemes exhibit a 5.2%~13.2% lower QoE under Sintel. However, the relative performance between all schemes remains consistent across different videos.

Reinjection control parameter β. A smaller value of β in Eq. 10 leads to Chorus reinjecting packets earlier during transmitting the chunk, which increases the number of reinjected packets. We evaluate Chorus using various values of β, ranging from 0 to 1.2, as depicted in Fig. 17. Here, $\beta = 0$ corresponds to RS + RI-U in Fig. 11a. It can be seen that as β increases from 0 to 0.9, the reinjection ratio steadily decreases, while the QoE performance remains relatively stable (within 2.7% difference). However, once β exceeds 1.0, indicating that Chorus reinjects after the chunk transmission time surpasses the expected time, the QoE performance experiences a rapid decline. Consequently, a suitable choice for Chorus to strike a balance between QoE improvement and reinjection cost is $\beta = 0.9$.

QoE frame sending period. To determine the optimal period that Chorus server sends QoE frame, we test a range of values in [10, 20, 50, 100, 200, 500, 1000, 2000, 3000, 4000, 6000]ms. Fig. 18 presents the QoE performance of different period settings. When the period is less than 100ms, QoE frames are sent too frequently. This excessive frequency can slightly disrupt chunk transmission and reduce QoE by 2.6%~6.9%. Specifically, given that the average chunk transmission time of Chorus in the emulation is 3.9s, a 10ms period will introduce an average of 390 QoE frames during

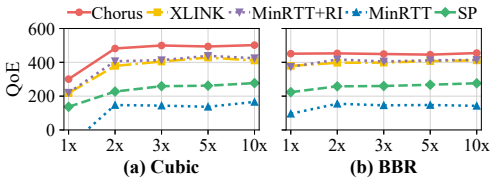


Figure 15: QoE of all schemes under different link buffer sizes (multiples of BDP) and CCAs.

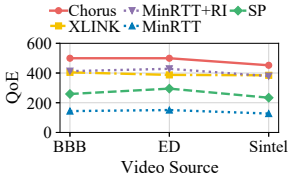


Figure 16: QoE of all under different source videos.

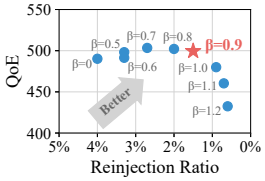


Figure 17: Reinjection parameter β of Chorus.

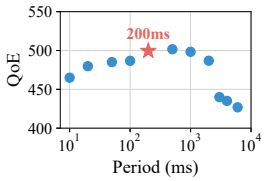


Figure 18: QoE sending period of Chorus server.

transmitting each chunk. On the other hand, infrequent sending of QoE frames (period greater than 2000ms) can lead to inaccurate throughput prediction on the client and further decrease QoE by 2.5%~14.6%. Therefore, a period of 200ms to 1000ms is most appropriate. We choose 200ms as the period for Chorus, in case of sudden network changes.

5.5 Real-world Evaluation

We finally evaluate Chorus on real-world mobile Internet.

Methodology. We conducted tests in three scenarios, categorized by the average bandwidth of the WiFi and cellular links: *(i) Strong Scenario*, where the WiFi link's average bandwidth surpasses the top bitrate (i.e., 16 Mbps), and all considered schemes should be able to persistently select the highest bitrate after the start-up phase. *(ii) Medium Scenario*, where the WiFi link's bandwidth is lower than the highest bitrate, but the cellular link's bandwidth is adequate. This common real-world situation [32] highlights the advantage of multipath schemes over SP. *(iii) Weak Scenario*, where the aggregated bandwidth of both links is below the video's highest bitrate. This scenario often occurs with mobility or weak wireless signal strength, thus exhibiting a high dynamic range of bandwidth, and is regarded as the heavy tail of real-world mobile Internet [50, 90, 96].

Since MinRTT performs the worst and MinRTT+RI performs similarly to XLINK (within 2% difference) in our emulation (§5.2), we only compared Chorus with the SP and XLINK for the convenience of collecting the dataset in the real world. For every scenario, three schemes underwent 12 tests. Half of the tests were conducted while stationary and the other half while walking. During the walking tests, handovers may occur between WiFi access points or cellular base stations. We considered various types of wireless links, including 2.4GHz/5GHz WiFi (WiFi 4, WiFi 5, or WiFi 6) and 4G/5G cellular. Although these experiments are inherently unrepeatable, we ran tests for each scenario several times under identical conditions and timings, striving to minimize the influence of uncontrollable factors. In total, we collected 108 real-world mobile sessions, each spanning 5 minutes. The average chunk throughput of all schemes under the three scenarios is 68.4 Mbps, 43.1 Mbps, and 5.5 Mbps, respectively.

Overall performance. Fig. 19 displays the overall performance of all considered schemes across all scenarios in real-world mobile Internet (with 95% confidence). The main observations align with the results of the emulation (§5.2). Compared to XLINK and SP, Chorus improves the total QoE by 65.7% and 114.4% on average, respectively. Specifically, Chorus achieves a similar bitrate sum with XLINK (18.8% higher than SP) and the lowest rebuffering time (48.6% and 39.2% reduction over XLINK and SP, respectively). Fig. 19b further shows that Chorus performs well in almost all tests.

Performance breakdown. A closer inspection of the results reveals that *Chorus's main improvement in QoE comes from the weak scenario*, as illustrated in Fig. 20. In both strong and medium scenarios, Chorus parallels XLINK's near-optimal QoE. Specifically, Chorus experiences no rebuffering after the first chunk and has the shortest total rebuffering time. Although SP maintains the highest bitrate selection when stationary in the strong scenario, its QoE decreases during mobility or in the medium scenario. Conversely, multipath schemes can leverage an extra link to maintain the highest video quality. In the weak scenario, Chorus notably outperforms XLINK and SP by reducing the rebuffering time by 48.1% and 33.7% on average, respectively. As seen in the emulation findings (§5.2), these gains stem from Chorus's enhanced throughput prediction and expected-time-oriented transport performance while minimizing costs.

Performance in the weak scenario. The results in weak scenarios are illustrated in Fig. 21. Note that due to high rebuffering time, all schemes present an average negative QoE. In most tests, XLINK performs the worst (Fig. 21a). Although SP performs well in several tests, its performance is quite unstable, sometimes hardly completing the test. In contrast, *Chorus provides the most stable performance in the worst cases and achieves the best QoE on average.* Fig. 21b further presents that Chorus outperforms XLINK on transport performance, in terms of average chunk throughput (6.9% increase), loss rate (55.6% decrease), and reinjection ratio (10.7% decrease). The performance issues of XLINK-based ABR originate from two primary factors. First, XLINK introduces additional uncertainty in chunk throughput prediction, leading to inappropriate (excessively high) bitrate selection, as analyzed in §2.3. Second, XLINK lacks a comprehensive understanding of the essential QoE requirements for adaptive streaming, resulting in an inability to meet them while balancing costs. These results demonstrate that Chorus excels in optimizing the QoE of the mobile Internet, especially in the heavy tail.

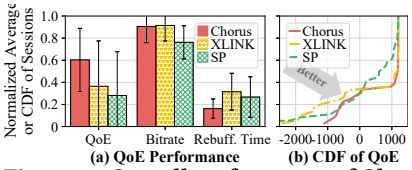


Figure 19: Overall performance of Chorus vs. baselines in the real world.

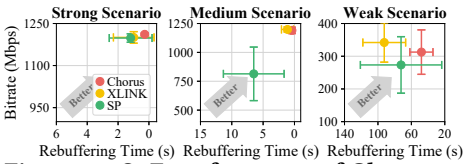


Figure 20: QoE performance of Chorus vs. baselines under various scenarios.

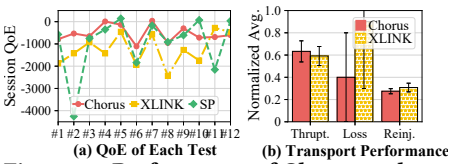


Figure 21: Performance of Chorus vs. baselines under real-world weak scenarios.

6 Limitations and Discussion

Better throughput prediction. While Chorus achieves outstanding QoE based on a simple predictor, its accuracy could be further improved by data-driven methods, as done in single-path streaming [50, 56, 79, 90]. To achieve relatively conservative predictions, we can consider the overestimation ratio in the loss function. This is left as future work.

Combined with other ABR algorithms. We evaluated Chorus with MPC [95] because it is well studied [1, 27, 50, 79, 90], widely applied [18, 45, 62, 78], and easy to deploy. Nonetheless, given that most latest ABR algorithms in VoD rely on throughput predictions in both academic research [50, 56, 90] and industry [1, 51, 76], Chorus can be combined with any of them as a general framework. Furthermore, Chorus's design also suits other HAS-based video applications, such as live video [45, 78], 360° video [18, 62], and volumetric video [47], by interacting with their ABR algorithms.

Other QoE metrics. Following previous works, we used the linear QoE definition proposed in MPC [95] because it directly corresponds to the ABR bitrate selection. Nevertheless, studies on QoE assessment have developed other QoE metrics to more precisely model the subjective or objective video quality and user perception, such as PSNR, SSIM [85], VMAF [44], and ITU-T Rec. P.1203/1204 [64, 67]. As research in this field is ongoing, we will investigate which metrics can aid Chorus in enhancing real user satisfaction in the future.

Reducing cellular usage. One concern of applying multipath transmission is cellular data consumption. In fact, Chorus is flexible enough to adapt to user preferences regarding data usage. In the CD phase, Chorus can limit cellular usage by assigning more packets (adjusting α in Eq. 3) on the WiFi link; in the FC phase, Chorus can reinject fewer packets on the cellular link, e.g., with a probability less than one.

Extension to multiple paths. Chorus is inherently scalable to handle multiple paths. In scenarios with more than two paths, Chorus can maintain and periodically update a list of all available paths, ranking them based on the latest bandwidth. Packet scheduling (assignment ratio) is then dynamically adjusted, ensuring optimal utilization of each path. For packet reinjection, Chorus chooses the available path (other than the packet's original path) with the shortest RTT.

7 Related Work

Numerous works have explored multipath video streaming. However, only a few are related to our work due to the vast design space encompassing multipath and video streaming. Many focus on multi-source video [11, 15] or non-transport layer multipath [4, 60, 61]. Other MPTCP/MPQUIC-based works either do not target HTTP-based adaptive streaming [17, 20, 59, 88, 98, 101], or concentrate on video coding [22, 92] or multipath congestion control [97].

DEMS [30], ECF [46], MP-DASH [32], and PERM [28] are most relevant to our work. However, DEMS and ECF solely optimize the packet scheduler without coordinating ABR logic. MP-DASH and PERM aim to maintain rather than optimize QoE while reducing cellular usage through cross-layer design, but their results do not include QoE improvements over MinRTT. In detail, MP-DASH primarily uses the WiFi link (not necessarily the best single path) for transmission, enabling multipath only when long transmission times are predicted. PERM employs a deep reinforcement learning model as its client-side scheduler and ABR algorithm, which suffers from delayed reactions to network changes [61] and limited computational capacity of mobile devices [94], making it challenging to deploy in practical systems.

8 Conclusion

Most previous studies on mobile multipath transmission aim to optimize transport performance. However, this optimization does not necessarily induce better QoE for adaptive streaming. The root cause is that adaptive streaming is uncoordinated with multipath scheduling. To this end, this paper proposes Chorus, a cross-layer coordination framework for multipath scheduling and ABR algorithms to optimize QoE jointly. Chorus incorporates two-way feedback control loops and introduces Coarse-grained Decisions and Fine-grained Corrections (CD&FC). In this way, Chorus ensures appropriate bitrate selection and expected-time-oriented transport performance for multipath adaptive streaming. Chorus is designed with fewer assumptions and is practical to deploy in mobile applications. Evaluations in emulation and real-world mobile Internet demonstrate Chorus's consistent superiority in optimizing QoE, especially in the heavy tail.

Acknowledgments

We thank the anonymous shepherd and reviewers for their constructive feedback. This work was supported in part by the National Key R&D Program of China (2022YFB2901800) Natural Science Foundation of China (U20A20180, 62072437), and Beijing NSF (JQ20024).

References

[1] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-Tuning Video ABR Algorithms to Network Conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 44–58. https://doi.org/10.1145/3230543.3230558

[2] alibaba/xquic. 2023. https://github.com/alibaba/xquic.

[3] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 329–342. https://www.usenix.org/conference/nsdi18/presentation/arun

[4] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. 2019. Jigsaw: Robust Live 4K Video Streaming. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 14, 16 pages. https://doi.org/10.1145/3300061.3300127

[5] Bento4. 2023. https://www.bento4.com/.

[6] Bilibili. 2018. Instructions on introducing DASH technology to improve user playback experience. https://www.bilibili.com/read/cv949156.

[7] Bitmovin. 2015. Why YouTube and Netflix use MPEG-DASH in HTML5. https://bitmovin.com/mpeg-dash-youtube-netflix-html5/.

[8] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. 2009. TCP Congestion Control. RFC 5681. https://doi.org/10.17487/RFC5681

[9] Olivier Bonaventure. 2015. Multipath TCP is pronounced GIGA Path. http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html.

[10] Olivier Bonaventure. 2018. Apple uses Multipath TCP. http://blog.multipath-tcp.org/blog/html/2018/12/15/apple_and_multipath_tcp.html.

[11] Joachim Bruneau-Queyreix, Mathias Lacaud, Daniel Negru, Jordi Mongay Batalla, and Eugen Borcoci. 2017. MS-Stream: A multiple-source adaptive streaming solution enhancing consumer's perceived quality. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 427–434. https://doi.org/10.1109/CCNC.2017.7983147

[12] Big Buck Bunny. 2023. https://peach.blender.org/.

[13] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[14] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, Ian Swett, and Van Jacobson. 2022. *BBR Congestion Control*. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/ Work in Progress.

[15] Yung-Chih Chen, Don Towsley, and Ramin Khalili. 2016. MSPlayer: Multi-source and multi-path video streaming. *IEEE Journal on Selected Areas in Communications* 34, 8 (2016), 2198–2206.

[16] MediaPlayer-Extended: Android MediaPlayer API compatible media player library with exact seek and DASH support. 2023. https://github.com/protyposis/MediaPlayer-Extended.

[17] Xavier Corbillon, Ramon Aparicio-Pardo, Nicolas Kuhn, Géraldine Texier, and Gwendal Simon. 2016. Cross-Layer Scheduler for Video Streaming over MPTCP. In *Proceedings of the 7th International Conference on Multimedia Systems* (Klagenfurt, Austria) *(MMSys '16)*. Association for Computing Machinery, New York, NY, USA, Article 7, 12 pages. https://doi.org/10.1145/2910017.2910594

[18] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R. Das. 2020. Streaming 360-Degree Videos Using Super-Resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. IEEE, 1977–1986. https://doi.org/10.1109/INFOCOM41043.2020.9155477

[19] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath QUIC: Design and Evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (Incheon, Republic of Korea) *(CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 160–166. https://doi.org/10.1145/3143361.3143370

[20] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2023. Converge: QoE-driven Multipath Video Conferencing over WebRTC. In *Proceedings of the ACM SIGCOMM 2023 Conference*. Association for Computing Machinery, New York, NY, USA, 637–653.

[21] Elephants Dream. 2023. https://orange.blender.org/.

[22] Anis Elgabli, Ke Liu, and Vaneet Aggarwal. 2018. Optimized preference-aware multi-path video streaming with scalable video coding. *IEEE Transactions on Mobile Computing* 19, 1 (2018), 159–172.

[23] Simone Ferlin, Özgü Alay, Olivier Mehani, and Roksana Boreli. 2016. BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. In *2016 IFIP networking conference (IFIP networking) and workshops*. IEEE, 431–439.

[24] FFmpeg. 2023. https://www.ffmpeg.org/.

[25] Alan Ford, Costin Raiciu, Mark J. Handley, and Olivier Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824. https://doi.org/10.17487/RFC6824

[26] Alexander Frommgen, Tobias Erbshäußer, Alejandro Buchmann, Torsten Zimmermann, and Klaus Wehrle. 2016. ReMP TCP: Low latency multipath TCP. In *2016 IEEE international conference on communications (ICC)*. IEEE, 1–7.

[27] Ehab Ghabashneh and Sanjay Rao. 2020. Exploring the interplay between CDN caching and video streaming performance. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 516–525.

[28] Yushuo Guan, Yuanxing Zhang, Bingxuan Wang, Kaigui Bian, Xiaoliang Xiong, and Lingyang Song. 2020. PERM: Neural adaptive video streaming with multi-path transmission. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1103–1112.

[29] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360° Video Streaming with a Better Understanding of Quality Perception. In *Proceedings of the ACM Special Interest Group on Data Communication*. Association for Computing Machinery, New York, NY, USA, 394–407. https://doi.org/10.1145/3341302.3342063

[30] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (Snowbird, Utah, USA) *(MobiCom '17)*. Association for Computing Machinery, New York, NY, USA, 141–153. https://doi.org/10.1145/3117811.3117829

[31] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[32] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies* (Irvine, California, USA) *(CoNEXT '16)*. Association for Computing Machinery, New York, NY, USA, 129–143. https://doi.org/10.1145/2999572.2999606

[33] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, and Lifeng Sun. 2022. Learning Tailored Adaptive Bitrate Algorithms to Heterogeneous Network Conditions: A Domain-Specific Priors and Meta-Reinforcement Learning Approach. *IEEE Journal on Selected Areas in Communications* 40, 8 (2022), 2485–2503. https://doi.org/10.1109/JSAC.2022.3180804

[34] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2020. Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1967–1976.

[35] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 187–198. https://doi.org/10.1145/2619239.2626296

[36] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. https://doi.org/10.17487/RFC9000

[37] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (Nice, France) *(CoNEXT '12)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/2413176.2413189

[38] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. 2020. Online Learning for Low-Latency Adaptive Streaming. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 315–320. https://doi.org/10.1145/3339825.3397042

[39] S. Shunmuga Krishnan and Ramesh K. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. In *Proceedings of the 2012 Internet Measurement Conference* (Boston, Massachusetts, USA) *(IMC '12)*. Association for Computing Machinery, New York, NY, USA, 211–224. https://doi.org/10.1145/2398776.2398799

[40] Nicolas Kuhn, Emmanuel Lochin, Ahlem Mifdaoui, Golam Sarwar, Olivier Mehani, and Roksana Boreli. 2014. DAPS: Intelligent delay-aware packet scheduling for multipath transport. In *2014 IEEE international conference on communications (ICC)*. IEEE, 1222–1227.

[41] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[42] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. 2018. RAVEN: Improving Interactive Latency for the Connected Car. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 557–572. https://doi.org/10.1145/3241539.3241571

[43] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. 2018. A Measurement Study on Multi-Path TCP with Multiple Cellular Carriers on High Speed Rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 161–175. https://doi.org/10.1145/3230543.3230556

[44] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, Megha Manohara, et al. 2016. Toward a practical perceptual video quality metric. *The Netflix Tech Blog* 6, 2 (2016), 2.

[45] May Lim, Mehmet N. Akcay, Abdelhak Bentaleb, Ali C. Begen, and Roger Zimmermann. 2020. When They Go High, We Go Low: Low-Latency Live Streaming in Dash.Js with LoL. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 321–326. https://doi.org/10.1145/3339825.3397043

[46] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (Incheon, Republic of Korea) *(CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 147–159. https://doi.org/10.1145/3143361.3143376

[47] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: Practical Mobile Volumetric Video Streaming through Multiview Transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking* (Sydney, NSW, Australia) *(MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 514–527. https://doi.org/10.1145/3495243.3517027

[48] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2023. *Multipath Extension for QUIC*. Internet-Draft draft-ietf-quic-multipath-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/05/ Work in Progress.

[49] Yanmei Liu, Yunfei Ma, Christian Huitema, Qing An, and Zhenyu Li. 2021. *Multipath Extension for QUIC*. Internet-Draft draft-liu-multipath-quic-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-liu-multipath-quic/04/ Work in Progress.

[50] Gerui Lv, Qinghua Wu, Weiran Wang, Zhenyu Li, and Gaogang Xie. 2022. Lumos: towards Better Video Streaming QoE through Accurate Throughput Prediction. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. IEEE, 650–659. https://doi.org/10.1109/INFOCOM48880.2022.9796948

[51] Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, and Eytan Bakshy. 2019. Real-world video adaptation with reinforcement learning. In *ICML 2019 Workshop RL4RealLife*.

[52] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[53] Streaming Media. 2014. Hulu: 'DASH Is Definitely the Future for Us'. https://www.streamingmedia.com/Articles/Editorial/Featured-Articles/Hulu-DASH-Is-Definitely-the-Future-for-Us-97468.aspx.

[54] 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 *Proceedings of the ACM SIGCOMM 2022 Conference* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 193–206. https://doi.org/10.1145/3544216.3544225

[55] mpshell. 2023. https://github.com/ravinet/mahimahi/tree/old/mpshell_scripted.

[56] Yun Seong Nam, Jianfei Gao, Chandan Bothra, Ehab Ghabashneh, Sanjay Rao, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2022. Xatu:

Richer Neural Network Based Prediction for Video Streaming. *SIGMETRICS Perform. Eval. Rev.* 50, 1 (jun 2022), 9–10. https://doi.org/10.1145/3547353.3522641

[57] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, Feng Qian, and Zhi-Li Zhang. 2021. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 610–625. https://doi.org/10.1145/3452296.3472923

[58] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. USENIX Association, Santa Clara, CA, 417–429. https://www.usenix.org/conference/atc15/technical-session/presentation/netravali

[59] Yunzhe Ni, Zhilong Zheng, Xianshang Lin, Fengyu Gao, Xuan Zeng, Yirui Liu, Tao Xu, Hua Wang, Zhidong Zhang, Senlang Du, et al. 2023. CellFusion: Multipath Vehicle-to-Cloud Video Streaming with Network Coding in the Wild. In *Proceedings of the ACM SIGCOMM 2023 Conference*. Association for Computing Machinery, New York, NY, USA, 668–683.

[60] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z. Morley Mao, and Subhabrata Sen. 2016. An In-Depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (New York City, New York) *(MobiCom '16)*. Association for Computing Machinery, New York, NY, USA, 189–201. https://doi.org/10.1145/2973750.2973769

[61] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z. Morley Mao. 2019. MP-H2: A Client-Only Multipath Solution for HTTP/2. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 10, 16 pages. https://doi.org/10.1145/3300061.3300131

[62] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 99–114. https://doi.org/10.1145/3241539.3241565

[63] Yanyuan Qin, Shuai Hao, K. R. Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2018. ABR Streaming of VBR-Encoded Videos: Characterization, Challenges, and Solutions. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (Heraklion, Greece) *(CoNEXT '18)*. Association for Computing Machinery, New York, NY, USA, 366–378. https://doi.org/10.1145/3281411.3281439

[64] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Erfurt, 1–6. https://doi.org/10.1109/QoMEX.2017.7965631

[65] Costin Raiciu, Mark J. Handley, and Damon Wischik. 2011. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356. https://doi.org/10.17487/RFC6356

[66] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX Association, San Jose, CA, 399–412. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/raiciu

[67] Rakesh Rao Ramachandra Rao, Steve Göring, Peter List, Werner Robitza, Bernhard Feiten, Ulf Wüstenhagen, and Alexander Raake. 2020. Bitstream-Based Model Standard for 4K/UHD: ITU-T P.1204.3 — Model Details, Evaluation, Analysis and Open Source Implementation. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX48832.2020.9123110

[68] YouTube recommended upload encoding settings. 2023. https://support.google.com/youtube/answer/1722171.

[69] Dash-Industry-Forum/dash.js: A reference client implementation for the playback of MPEG DASH via Javascript and compliant browsers. 2023. https://github.com/Dash-Industry-Forum/dash.js/.

[70] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. 2019. MuSher: An Agile Multipath-TCP Scheduler for Dual-Band 802.11ad/Ac Wireless LANs. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 34, 16 pages. https://doi.org/10.1145/3300061.3345435

[71] SANDVINE. 2023. 2023 Global Internet Phenomena Report. https://www.sandvine.com/global-internet-phenomena-report-2023.

[72] Golam Sarwar, Roksana Boreli, Emmanuel Lochin, Ahlem Mifdaoui, and Guillaume Smith. 2013. Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer. In *2013 27th international conference on advanced information networking and applications workshops*. IEEE, 1119–1124.

[73] The Tengine Web Server. 2023. https://tengine.taobao.org/.

[74] Hang Shi, Yong Cui, Xin Wang, Yuming Hu, Minglong Dai, Fanzhao Wang, and Kai Zheng. 2018. STMS: Improving MPTCP Throughput Under Heterogeneous Networks. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 719–730. https://www.usenix.org/conference/atc18/presentation/shi

[75] Sintel. 2023. https://durian.blender.org/.

[76] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 123–137. https://doi.org/10.1145/3204949.3204953

[77] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[78] Liyang Sun, Tongyu Zong, Yong Liu, Yao Wang, and Haihong Zhu. 2019. Optimal strategies for live video streaming in the low-latency regime. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 1–4.

[79] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 272–285. https://doi.org/10.1145/2934872.2934898

[80] Christian Timmerer, Matteo Maiero, and Benjamin Rainer. 2016. Which Adaptation Logic? An Objective and Subjective Performance Evaluation of HTTP-based Adaptive Media Streaming Systems. arXiv:1606.00341 [cs.MM]

[81] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 2405–2413. https://doi.org/10.1145/3343031.3350917

[82] Santiago Vargas, Rebecca Drucker, Aiswarya Renganathan, Aruna Balasubramanian, and Anshul Gandhi. 2021. BBR Bufferbloat in DASH Video. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 329–341. https://doi.org/10.1145/3442381.3450061

[83] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. 2018. Multipath QUIC: A deployable multipath transport protocol. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[84] Shibo Wang, Shusen Yang, Hailiang Li, Xiaodan Zhang, Chen Zhou, Chenren Xu, Feng Qian, Nanbin Wang, and Zongben Xu. 2022. SalientVR: Saliency-Driven Mobile 360-Degree Video Streaming with Gaze Information. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking* (Sydney, NSW, Australia) *(MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 542–555. https://doi.org/10.1145/3495243.3517018

[85] 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.

[86] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. USENIX Association, Lombard, IL, 459–471. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein

[87] Hongjia Wu, Özgü Alay, Anna Brunstrom, Simone Ferlin, and Giuseppe Caso. 2020. Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2295–2310.

[88] Jiyan Wu, Chau Yuen, Bo Cheng, Ming Wang, and Junliang Chen. 2015. Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing* 15, 9 (2015), 2345–2361.

[89] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. 2016. PiStream: Physical Layer Informed Adaptive Video Streaming Over LTE. *GetMobile: Mobile Comp. and Comm.* 20, 2, 31–34. https://doi.org/10.1145/3009808.3009819

[90] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 495–511. https://www.usenix.org/conference/nsdi20/presentation/yan

[91] Fan Yang, Qi Wang, and Paul D Amer. 2014. Out-of-order transmission for in-order arrival scheduling for multipath TCP. In *2014 28th international conference on advanced information networking and applications workshops*. IEEE, 749–752.

[92] Wang Yang, Jing Cao, and Fan Wu. 2021. Adaptive Video Streaming with Scalable Video Coding using Multipath QUIC. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 1–7.

[93] yangqingyuan/TekiXquic. 2023. https://github.com/yangqingyuan/TekiXquic.

[94] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. NEMO: Enabling Neural-Enhanced Video Streaming on Commodity Mobile Devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages. https://doi.org/10.1145/3372224.3419185

[95] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 325–338. https://doi.org/10.1145/2785956.2787486

[96] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. 2021. Loki: Improving Long Tail Performance of Learning-Based Real-Time Video Adaptation by Fusing Rule-Based Models. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 775–788. https://doi.org/10.1145/3447993.3483259

[97] Jia Zhao, Jiangchuan Liu, Cong Zhang, Yong Cui, Yong Jiang, and Wei Gong. 2020. MPTCP+: Enhancing Adaptive HTTP Video Streaming over Multipath. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–6.

[98] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, Qing An, Hai Hong, Hongqiang Harry Liu, and Ming Zhang. 2021. XLINK: QoE-Driven Multi-Path QUIC Transport in Large-Scale Video Services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 418–432. https://doi.org/10.1145/3452296.3472893

[99] 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 *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 29, 16 pages. https://doi.org/10.1145/3300061.3345430

[100] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. 2015. Can Accurate Predictions Improve Video Streaming in Cellular Networks?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (Santa Fe, New Mexico, USA) *(HotMobile '15)*. Association for Computing Machinery, New York, NY, USA, 57–62. https://doi.org/10.1145/2699343.2699359

[101] Xutong Zuo, Yong Cui, Xin Wang, and Jiayu Yang. 2022. Deadline-aware Multipath Transmission for Streaming Blocks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2178–2187.