AUGUR: Practical Mobile Multipath Transport Service for Low Tail Latency in Real-Time Streaming

Yuhan Zhou, *School of Computer Science, Peking University and Tencent Inc.;* Tingfeng Wang, *Tencent Inc.;* Liying Wang, *School of Computer Science, Peking University;* Nian Wen, Rui Han, Jing Wang, Chenglei Wu, Jiafeng Chen, and Longwei Jiang, *Tencent Inc.;* Shibo Wang, *Xi'an Jiaotong University and Tencent Inc.;* Honghao Liu, *Tencent Inc.;* Chenren Xu, *School of Computer Science, Peking University and Zhongguancun Laboratory and Key Laboratory of High Confidence Software Technologies, Ministry of Education* (*PKU*)

https://www.usenix.org/conference/nsdi24/presentation/zhou-yuhan

This paper is included in the Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation.

April 16–18, 2024 • Santa Clara, CA, USA

978-1-939133-39-7

Open access to the Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation is sponsored by

AUGUR: Practical Mobile Multipath Transport Service for Low Tail Latency in Real-Time Streaming

Yuhan Zhou[*ℙ𝕋], *Tingfeng Wang*[*𝕋], *Liying Wang*[ℙ], *Nian Wen*[𝕋], *Rui Han*[𝕋], *Jing Wang*[𝕋]
Chenglei Wu[𝕋], *Jiafeng Chen*[𝕋], *Longwei Jiang*[𝕋], *Shibo Wang*[𝕏𝕋], *Honghao Liu*[𝕋†], *Chenren Xu*[ℙℤ𝕂†]
[ℙ]*School of Computer Science, Peking University*
[𝕋]*Tencent Inc.* [𝕏] *Xi'an Jiaotong University* [ℤ]*Zhongguancun Laboratory*
[𝕂]*Key Laboratory of High Confidence Software Technologies, Ministry of Education (PKU)*

Abstract – Real-time streaming applications like cloud gaming require consistently low latency, even at the tail. Our large-scale measurement based on a major cloud gaming service provider reveals that in Wi-Fi networks, the delay of the wireless hop can inflate due to its fluctuating nature, making it difficult to achieve consistently low tail latency. While cellular paths can be leveraged to alleviate the impact of wireless fluctuation of Wi-Fi paths, our user study reveals that it is crucial to constrain cellular data usage while using multipath transport. In this paper, we present AUGUR, a multipath transport service designed to reduce long tail latency and video frame stall rates in mobile real-time streaming. To address the challenge of reducing long tail latency by utilizing cellular paths while minimizing cellular data usage, AUGUR captures user characteristics by deriving state probability models and formulates the equilibrium into Integer Linear Programming (ILP) problems for each user session to determine the opportunity of frame retransmission and path selection. Our trace-driven emulation and large-scale real-world deployment in Tencent Start cloud gaming platform demonstrate that AUGUR achieves up to 66.0% reduction in tail latency and 99.5% reduction in frame stall rate with 88.1% decrease in cellular data usage compared to other multipath transport schemes.

1 Introduction

Emerging real-time streaming applications like cloud gaming [1, 2], video conferencing [3, 4], and AR/VR [5, 6] provide users with interactive experiences for both entertainment and business. Such applications have grown rapidly worldwide and made a large market (*e.g.,* The global cloud gaming market reached $1.28 billion in 2022 and expects to reach $13.6 billion by 2028 [7]). However, to provide users with a seamless interactive experience, service providers must achieve *consistently* low tail latency [8]. Based on experience and statistics from our real-time streaming service platform, we observe that a tail latency (*i.e.,* 99.9th percentile) of 200 ms can lead to frequent video frame stalls, and even a 0.5% increase in stall rate results in a 33% drop in user retention time (§2.2). Therefore, it is essential to reduce long tail latency to improve user experience.

However, for mobile devices that access wireless networks, existing solutions for latency-intensive applications fail to meet the consistently low tail latency requirements imposed by interactive real-time streaming or are impractical to be widely deployed. The most intuitive and practical method to eliminate high network latency is to deploy a dedicated congestion control algorithm (CCA). However, we observe that with Wi-Fi networks, path RTT inflation caused by random wireless fluctuations can occur. Therefore, while existing CCAs designed for real-time streaming [9, 10, 11, 12] and wireless networks [13, 14] can provide sufficient bandwidth and low median latency, they fail to consistently achieve low tail latency (§2.2). Even with a low sending bitrate (*i.e.,* 512 Kbps), the tail latency can dramatically exceed 200 ms. Besides, RTT inflation is intrinsic in wireless links. Thus, it is difficult to achieve low tail latency with only one network path. Therefore, leveraging multipath is a straightforward approach to alleviate the impact of an RTT-fluctuating path.

Although many works on mobile multipath transport have been proposed to reduce latency by leveraging both Wi-Fi and cellular paths, they are not practical to be widely deployed to reduce long tail latency for real-time streaming applications in terms of performance, cellular cost, and deployability: *i)* Most multipath schedulers only make decisions on packet departure, leaving packet retransmission to the underlying loss recovery mechanism, thus the packets would be inevitably delayed in the presence of random RTT inflation [15, 16, 17, 18, 19, 20, 21]. Therefore they cannot meet the performance requirement of consistently low tail latency; *ii)* While most multipath transport schemes only focus on the performance heterogeneity of multiple paths, they ignore the data budget heterogeneity of Wi-Fi and cellular paths [17, 22]. Since Internet service providers (ISPs) provide cellular connections with limited data budget [23], we find that users of our mobile real-time applications express strong concerns about cellular data usage (§2.3). Therefore, a multipath transport system that consumes a large amount of cellular data is not practical to be widely deployed; *iii)* Many existing multipath transport systems require modification to

*equal contribution.
†corresponding authors.
✉: coreyliu@tencent.com ✉: chenren@pku.edu.cn

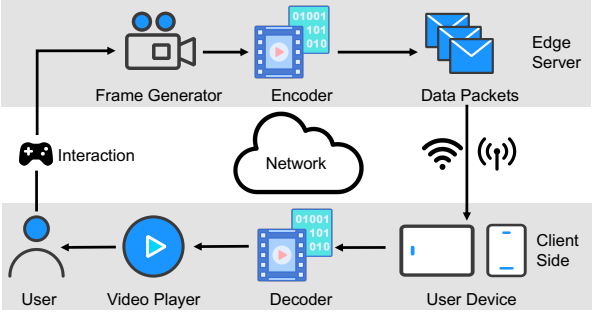


Figure 1: Overview of the real-time streaming pipeline.

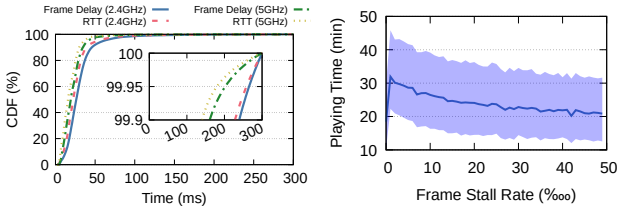


Figure 2: Long tail latency in real-time streaming.

Figure 3: Correlation between video frame stall rate and user retention time.

the kernel of user devices [16, 17, 18, 19] or rely on values that do not exist in real-time streaming (*e.g.,* volume video size [21] and playback buffer occupancy [24]) for decision-making. Consequently, they are infeasible to be widely deployed for real-time streaming applications.

In this work, we propose a multipath transport service AUGUR in mobile real-time streaming that meet the requirement of reducing long tail latency, minimizing cellular data usage, and enabling large-scale deployment. AUGUR uses the Wi-Fi path as the primary path for frame transmission and leverages the cellular path by introducing *application-level frame retransmission* and *primary path switch scheduling*. It tackles two challenges when leveraging multipath to deal with an RTT-fluctuating path: *i)* the RTT inflation caused by wireless fluctuation of the Wi-Fi path is highly unpredictable and *ii)* the usage of cellular paths is strictly limited and should be minimized as much as possible. These two challenges make it difficult to determine when to use the cellular path to alleviate the impact of RTT inflation.

To address these challenges, we observe that while user characteristics such as frame stall rate and the impact of prolonged latency vary greatly among different users, they remain stable for an individual user over a certain period of time (§2.4). Based on this observation, AUGUR divides the statistics of a user session into states and creates *per-session probability models* for each user to quantify the various user characteristics. It then formulates the equilibrium of reducing tail latency by leveraging cellular path and constraining cellular data usage as Integer Linear Programming (ILP) problems and uses the state probability models as inputs. By solving the ILP problems, AUGUR determines the appropriate moments to utilize cellular paths to achieve both the performance requirement of consistently low latency and the cost requirement of minimizing cellular usage. To achieve the deployability requirement, AUGUR is fully implemented in edge servers and requires no modification to user devices, customed hardware, or network middleboxes, making it immediately deployable.

We have deployed AUGUR in Tencent Start [25] cloud gaming platform for over six months with a wide range of millions of users, resulting in tens of millions of hours of user retention time. Our trace-driven emulation and large-scale production experiments show that AUGUR can reduce up to 66.0% tail latency and 99.5% frame stall rate with 2.7% average data usage on the cellular path, compared to other multipath transport schemes.

Contributions.

- We conduct a large-scale and in-depth statistical analysis to demonstrate the characteristics of long tail latency in mobile real-time streaming applications within a production environment;
- We propose AUGUR, a multipath transport service designed to reduce the long tail latency and frame stall while maintaining a strictly limited cellular data usage;
- We deploy AUGUR in Tencent Start cloud gaming platform and demonstrate that it can significantly reduce tail latency and stall rate while incurring negligible cellular costs.

Ethical claim. All user feedback and data statistics collected in this work are obtained with explicit permission from the users and are anonymized to protect their privacy. This work does not raise any ethical concerns and conforms to the IRB policies of the authors' institutions.

2 Background and Motivation

In this section, we introduce the background of interactive real-time streaming (§2.1). We then collect real-world statistics to present the long tail latency caused by RTT inflation in Wi-Fi path transmission (§2.2) and the limitation of existing mobile multipath transport schemes (§2.3). Finally, we discuss the user characteristics observed from our online measurement (§2.4).

2.1 Interactive Real-Time Streaming

An interactive real-time streaming service like cloud gaming provides users with an interactive video experience. As shown in Fig. 1, a typical real-time streaming pipeline consists of four components: frame generator, codec (encoder and decoder), transport system, and video player. The encoder encodes the video frames produced by the frame generator, and each frame is typically partitioned into multiple data packets for network transport. After being delivered to the user devices through the network transport system, the frame is immediately decoded and displayed by

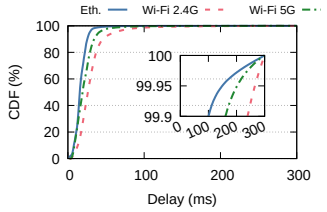
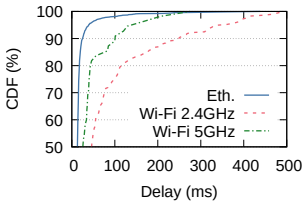
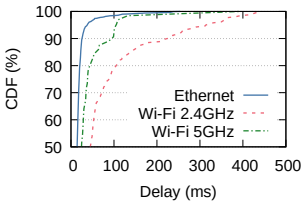


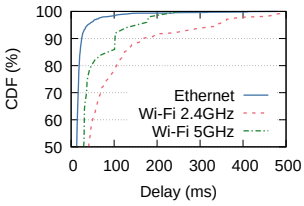
Figure 4: Frame delivery latency of wired and wireless networks.



(a) 512 Kbps.



(b) 1 Mbps.



(c) 2 Mbps.

Figure 5: Frame delivery latency over 99th percentile under low sending bitrate.

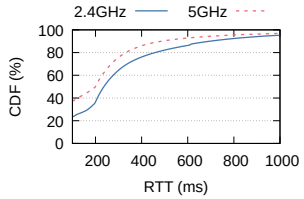


Figure 6: RTT inflation of Wi-Fi link before frame stall occurs.

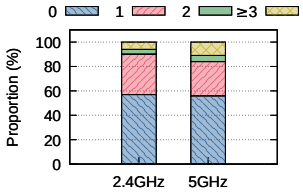


Figure 7: Packet loss rate on frame stalls with Wi-Fi link.

the video player without buffering. According to the inter-dependencies of video frames [26], all packets in a frame should be sent to the decoder to prevent video quality degradation. Furthermore, the user continuously interacts with the server, providing instructions or feedback to generate new video frames. This pipeline contrasts with buffer-based streaming applications such as video-on-demand (VoD), where the entire video content is pre-recorded, and a playback buffer in the video player can be utilized to absorb frame delivery latency [27] or improve user Quality of Experience (QoE) [24]. Therefore, to ensure a smooth and seamless interactive experience, it is crucial for the network transport system to guarantee consistently low latency for the timely delivery of all video frames, especially under fluctuating wireless network environments.

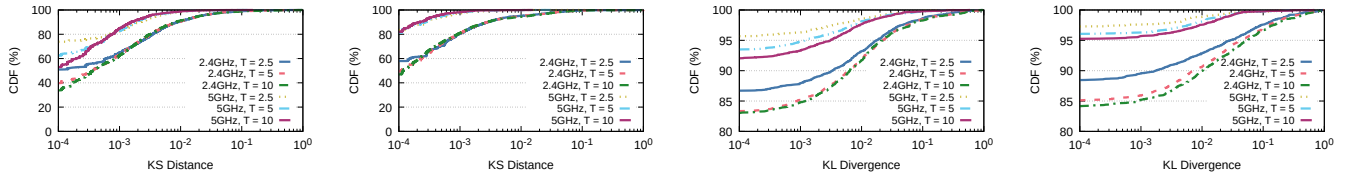
2.2 Long Tail Latency in Real-time Streaming

Long tail latency severely degrades user experience. The low latency requirement for frame delivery in real-time streaming means that even a slight increase in tail latency can have a significant impact on user experience. To investigate the impact of long tail latency, we conduct a measurement on our online cloud gaming service. As shown in Fig. 2, while the median RTT and frame delivery latency is kept below 30 ms, the tail latency (*i.e.,* 99.9th percentile) can reach over 200 ms, and such a severely delayed frame is highly likely to cause a video stall [8]. Moreover, as shown in Fig. 3, we observe that an increased video frame stall rate results in a significant decrease in user retention time, even an 0.5% increased stall rate would result in 33% drop in retention time, and there still exists an increase of user churn even the stall rate is kept below 0.1%, indicating user dissatisfaction with the service. Therefore, long tail latency severely degrades user experience, and achieving *consistently low latency* necessitates the attainment of at least a 99.9% in-time delivery of frames.

RTT inflation contributes to long tail latency instead of network congestion. Intuitively, packet loss and retransmission caused by network congestion significantly contribute to packet delivery latency [28]. However, with our private CCA designed for real-time streaming and edge servers deployed, we find that long tail latency is mainly induced by inflated path RTT due to wireless fluctuation for mobile devices. As shown in Fig. 4, compared to the wired network environment, when streaming through the wireless network connection from a Wi-Fi access point (AP) to mobile devices, the frame delivery latency at the 99th percentile is increased by up to 290%. Additionally, we plot the RTT of the wireless path *before* a frame stall occurs, Fig. 6 shows that the path RTT is already inflated to more than 200 ms for 65% of stalled frames on Wi-Fi 2.4G networks and 50% of stalled frames on Wi-Fi 5G, respectively. Furthermore, we observe that packet loss is infrequent on frame stalls. To demonstrate this, we plot the packet loss rate when frame stall happens and show the result in Fig. 7, nearly 60% of packets are delivered without loss during a frame stall. This indicates that the long tail latency is induced by the fluctuating intrinsic of the wireless path instead of network congestion.

CCAs fail to reduce latency at the tail. The most widely used method to reduce frame delivery latency in real-time streaming is to employ sophisticated CCAs. However, with inflated RTT caused by wireless fluctuation, while state-of-the-art CCAs designed for real-time streaming [9, 10, 11, 12] or wireless networks [13, 14] can reach a low median latency, they fail to ensure a consistently low tail latency. The primary goal of CCAs is to adapt to link capacity variations and promptly reduce the sending rate to eliminate queuing delay or packet loss to maintain low transport latency [29, 30, 31, 32]. Nonetheless, due to RTT inflation caused by Wi-Fi link fluctuation, the link is unable to deliver data in time. Consequently, the long tail latency cannot be addressed by deploying CCAs.

To demonstrate the limitations of CCAs under wireless link fluctuations, we conduct an experiment by enforcing



(a) P(frame stall) KS distance. (b) P(frame stall | RTT $> X$) KS distance (c) P(frame stall) KL divergence. (d) P(frame stall | RTT $> X$) KL divergence.

Figure 8: Kolmogorov-Smirnov distance and Kullback-Leibler divergence between the characteristics probability distributions of the current minute and the past T minutes during each individual user's playing time.

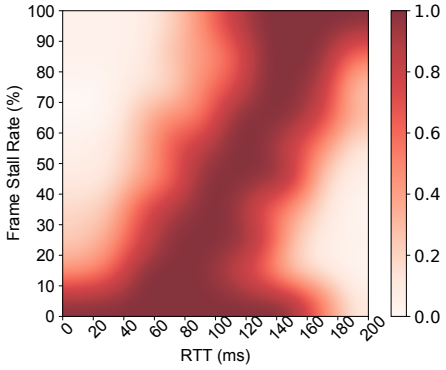


Figure 9: Overlayed curves of the relationship between frame stall rate and path RTT. Each point in each curve contributes to a point in the heatmap and the color indicates the number of points normalized by maximum.

our CCA to maintain extremely low sending bitrates that are below our video quality requirement of service (*i.e.,* 2 Mbps) and access our platform from both wired and wireless networks. As shown in Fig. 5, even with low sending bitrates, frame delivery latency at the tail remains high, and the latency distributions remain consistent across all low bitrates. This highlights that under fluctuating wireless networks, CCAs are insufficient to reduce latency at the tail for real-time streaming applications. Therefore, it is valuable to leverage cellular paths to alleviate the impact of wireless fluctuation of the Wi-Fi path.

2.3 Limitation of Existing Multipath Transport in Real-time Streaming

Multipath transport is a promising and practical solution to reduce long tail latency in mobile real-time streaming because: *i)* Wireless fluctuations are hardly correlated between heterogeneous wireless networks like Wi-Fi and cellular networks, indicating that prolonged latency induced by one path could be eliminated by another [17]; *ii)* Contemporary mobile devices are typically equipped with multiple network interfaces, enabling the large-scale deployment of a multipath transport service [24]. However, existing multipath transport schemes are not practical to be widely deployed to reduce tail latency for mobile real-time streaming in terms of performance, cellular cost, and deployability.

Multipath schedulers fail to achieve low tail latency. Although many works on multipath transport have been proposed, they mainly focus on maximizing throughput [15] or minimizing request completion time (RCT) [19, 20, 24]. Some multipath transport systems designed for latency-sensitive applications like BLEST [18] and RAVEN [17] schedule packet transmission on multiple paths but leave retransmission to the underlying loss recovery mechanism. However, packets scheduled on the Wi-Fi path could already be severely delayed before retransmission timeout (RTO) due to highly unpredictable RTT inflation, thus they cannot meet the performance requirement of low tail latency. XLINK [24] uses packet re-injection to reduce frame delivery latency, but it assumes a known video chunk size and requires playback buffer occupancy level as QoE signal, which cannot be used in real-time streaming.

Minimizing cellular data usage is crucial for practical deployment. Multipath schedulers designed for interactive applications like RAVEN [17] and ReMP [33] utilize cellular paths for redundant frame transmission but are unaware of cellular data cost. However, video frame delivery consumes a large amount of bandwidth and can lead to significant cellular data usage. To investigate users' concerns about cellular data cost, we explicitly deliver a questionnaire within our cloud gaming application to 1,251,420 users. All user feedback and data statistics collected are obtained with explicit permission and are anonymized to protect their privacy. Our analysis of online users' feedback reveals: *i)* While accessing our gaming platform with mobile devices, 89.8% of users prefer Wi-Fi connection due to the lower cost compared to cellular networks; *ii)* 57.0% of users are not willing to consume cellular data for streaming service; *iii)* For the remaining 43.0% of users willing to use cellular networks for better performance, 62.9% of them express a strong desire to reduce cellular data costs Therefore, we argue that for a practical multipath transport service, the cellular cost constraint should not be an incidental concern, but rather a primary design consideration. Although some multipath transport schemes like MP-DASH [21] and COM [34] reduce cellular data usage for VoD applications, they require the size of the pre-recorded video content, thus cannot be

used in real-time streaming.

Kernel-based multipath transport schemes are infeasible to widely deploy. Many existing multipath transport schemes require modifications to the kernel of user devices [16, 17, 18, 19, 21]. As a cloud gaming service provider, it is not feasible to make modifications to user devices, network middleboxes, or customed hardware. Therefore, they are impractical to be widely deployed.

2.4 User Characteristics in Real-time Streaming

Network characteristics vary among different users. Our real-time streaming platform is utilized by millions of users across a wide range of regions with different user device models and network environments. According to our measurements, there is a significant characteristics variation among users. We study the correlation between the frame stall rate and path RTT from different user sessions in our measurement. We plot the conditional probability $P(\text{frame stall} \mid \text{RTT} > X)$ curves and overlay all curves to formulate a heatmap. As shown in Fig. 9, while higher latency does increase the severely-tail rate overall, the correlation between them varies among different users. For example, when path RTT reaches 100 ms, the probability of a frame stall can range from 0% to 100% for different users. This indicates that RTT inflation has different impacts on different users and such variation in user characteristics should be considered for a widely deployed system.

Network characteristics remain stable for individual users. Although there is significant variation among user characteristics, we observe that there is some stability of statistics for an individual user over a certain period of time. While characteristics such as frame stall rate and network latency pattern can vary significantly among different users, they tend to remain relatively stable for an individual user over a time window. To demonstrate the stability, we computed the Kolmogorov-Smirnov (KS) distance [35] and Kullback-Leibler (KL) divergence [36] between the probability distributions of two characteristics, namely $P(\text{frame stall})$ and $P(\text{frame stall} \mid \text{RTT} > X)$, for both the current minute and the previous T minutes of each individual user's playing time and plot the CDFs of both the KS distance and KL divergence values for over 3000 users in Fig. 8. The results indicate that nearly 99% of the current minute's characteristics probability distributions are similar to those of the previous time window (*i.e.,* KS distance and KL divergence values are less than 0.1), suggesting a strong correlation between statistics calculated from a previous time window and the current minute. We leverage such stability to capture each user's characteristics in our design.

3 AUGUR Design

In this section, we propose a multipath transport service AUGUR to reduce the long tail latency induced by wireless link RTT inflation while minimizing cellular data usage. We first present the design goals of AUGUR.

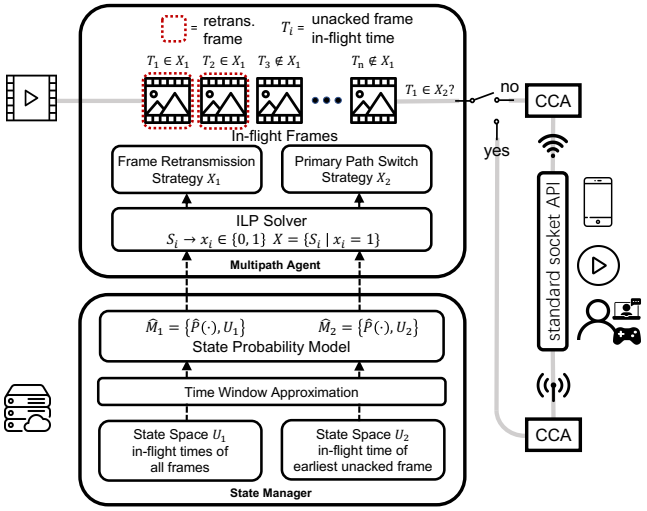


Figure 10: System overview of AUGUR.

3.1 Design Goals

The main idea behind AUGUR is to use the Wi-Fi path as the primary path to stream video frames and leverage the cellular network as a backup path when the Wi-Fi connection suffers wireless fluctuations. In light of the observation presented in §2, we design AUGUR to achieve three goals.

G1: Reduce long tail latency induced by wireless fluctuation. As presented in §2.2, the long tail latency resulting from Wi-Fi wireless fluctuation like RTT inflation is an inherent issue of the Wi-Fi path, thus it is valuable to also utilize the cellular path for data streaming. However, it is challenging to determine the opportunity of using another path because the RTT inflation of the Wi-Fi path is highly unpredictable.

G2: Constrain cellular data usage. One straightforward approach to leveraging a redundant path would be to replicate all frames on cellular paths, but this would incur a significant consumption of cellular data and violates the user concerns in §2.3. Moreover, with the significant imbalance in link capacity [37], the cellular link may be unable to handle the additional traffic from replicating all frames (*e.g.,* cellular bufferbloat [38]). Therefore AUGUR must carefully constrain the usage of the cellular path.

G3: Enable large-scale deployment. Our real-time streaming platform provides service to millions of users with various device models, operating systems, and Wi-Fi AP models. As a streaming service provider, it is not feasible to make modifications to user device kernels, network middleboxes, or customized hardware. To be practically deployed on a large scale, AUGUR should only require modifications to game servers and user applications.

3.2 System Overview

To address the challenge of reducing long tail latency by utilizing cellular paths while adhering to a strict cellular data usage constraint, AUGUR first leverages our observation on user characteristics (§2.4) and derives *per-session probability models* to capture the characteristics of each user. It then formulates the equilibrium of tail latency reduction and cellular cost minimization into Integer Linear Programming (ILP) problems. Using the measured probability models as inputs, AUGUR solves ILP problems to determine the opportunity to use the cellular path. As a multipath transport service, AUGUR leverages the cellular path through two approaches: *i) application-level frame retransmission.* AUGUR monitors in-flight frames sent on the Wi-Fi path and promptly retransmits frames that are likely to be severely delayed on the cellular path; *ii) primary path switch scheduling.* AUGUR transiently switches the primary path to cellular and streams all newly generated video frames through it in the presence of severe Wi-Fi path capacity degradation. As illustrated in Fig. 10, AUGUR consists of two main components: the *state manager* (§3.5) monitors the capacity of the Wi-Fi and cellular path and approximates probability models for each user. Based on the models provided by the state manager, the *multipath agent* (§3.6) continuously solves the formulated ILP problems to obtain cellular path utilization strategies, and performs frame retransmission and primary path switch.

In the rest of this section, we first introduce our state probability model for real-time streaming in §3.3 and our strategy for utilizing the backup path in §3.4. We then demonstrate how we apply our theoretical analysis into practice to derive the model (§3.5) and make decisions on frame retransmission and primary path switch(§3.6).

3.3 State Probability Model

In a real-time streaming session, a sender and a receiver communicate over a network path along a timeline. The sender continuously sends data, such as video frames, to the receiver. Concurrently, the receiver provides feedback (ACKs) to the sender regarding the data's arrival or at specific time intervals. The sender can then derive some statistics based on this feedback (*e.g.,* RTT, acknowledged bytes). To effectively utilize these statistics and extract vital information for the sender to determine the transmission pattern, we divide them into several *states*. We let $U = \bigcup_i \{S_i\}, \forall i \neq j, S_i \cap S_j = \emptyset$ donate all possible states. For example, if the sender chooses to use `(RTT, Bandwidth)` as states, U would be a two-dimensional space and each point in the space represents a state. For each session, the probability distribution of states $P(S_i)$ could be significantly different due to the great variety of user characteristics.

When the receiver experiences a frame stall, the next feedback can carry information about the stall event, allowing the sender to record it. If we define $stall$ to be the stall events across all states of a session, the overall frame stall probability for a user session would be:

$$P(stall) = P(stall, U) = \sum_{S_i \in U} P(stall, S_i)$$
$$= \sum_{S_i \in U} P(stall \mid S_i) P(S_i) \quad (1)$$

The distributions of $P(stall)$, $P(stall \mid S_i)$, and $P(S_i)$ can provide insight into the condition of the receiver and the network for a given user session. We define the state probability model for a user session as $M = \{P(stall), P(stall \mid S_i), P(S_i)\}, S_i \in U$, and the relationship between these distributions is given in equation Eqn. 1. By obtaining M, we can capture and quantify the user characteristics of a session.

3.4 Backup Path Utilization Strategy

To reduce long tail latency and frame stalls of the primary path, the sender can utilize a backup path for frame transmission. A backup path utilization strategy decides when the backup path should be used. Ideally, a backup path should only be used when frames are likely to be delayed on the primary path. However, it is difficult to accurately predict if a frame would be delayed and cause a stall before it is acknowledged. To effectively use the backup path, we break down the problem of deriving an optimal utilization strategy into two tasks:

T1: Covering delayed frames caused by wireless fluctuation. The backup path utilization strategy should minimize the false negative (FN) rate of the decisions by ensuring that it utilizes the backup path to cover delayed frames to the greatest extent possible.

T2: Reducing the utilization of backup paths. The backup path should only be used when necessary, as excessive utilization of the backup path can result in additional data usage and negative impacts (*e.g.,* increased cellular data charges in our case). The strategy should aim to minimize the false positive (FP) rate of the decisions.

As explained in §3.3, we divide the sender statistics of a real-time streaming process into states. We use these states as decision points. For each state S_i, we employ x_i to indicate whether the backup path should be utilized for frame transmission to mitigate delivery delay, where $x_i = 1$ denotes a positive decision and $x_i = 0$ denotes a negative decision. A *strategy* refers to a selection of several states from U, where the chosen states are represented as $X = \{S_i \in U \mid x_i = 1\}$, and the remaining states are represented as $\bar{X} = \{S_i \in U \mid x_i = 0\}$. We next derive the optimal backup path utilization strategy based on the state probability model M.

To achieve **T1**, we present the false negative rate of strat-

egy X to be:

$$P(FN) = P(\bar{X} \mid stall) = \frac{P(stall, \bar{X})}{P(stall)} = \frac{P(stall) - P(stall, X)}{P(stall)} = \frac{P(stall) - \sum_{S_i \in X} P(stall, S_i)}{P(stall)} = \frac{P(stall) - \sum_{S_i \in X} P(stall \mid S_i) P(S_i)}{P(stall)} \quad (2)$$

As discussed in §2.4, the stall rate of a single user session tends to remain stable in a time window. Therefore, we assume that in a certain period of time, the overall frame stall probability $P(stall)$ is a constant value. In this case, the minimization of $P(FN)$ turns into the maximization of function:

$$f(X) = \sum_{S_i \in X} P(stall \mid S_i) P(S_i) \quad (3)$$

This maximization function implicitly assumes that the capacity of the backup path is adequate for mitigating prolonged latency, which may not always be the case. Therefore, we introduce $P_{backup}(S_i)$ to represent the probability that the backup path is sufficient to perform frame transmission at state S_i. The maximization function in Eqn. 3 is modified as follows:

$$F(X) = \sum_{S_i \in X} P(stall \mid S_i) P(S_i) P_{backup}(S_i) \quad (4)$$

To achieve **T2**, we present the false positive rate of strategy X to be:

$$P(FP) = P(X \mid !stall) = \frac{P(X, !stall)}{P(!stall)} = \frac{P(X) - P(X, stall)}{1 - P(stall)} = \frac{P(X) - f(X)}{1 - P(stall)} \quad (5)$$

Given that the primary objective of **T1** is to maximize $f(X)$ while assuming a constant value of $P(stall)$, our aim is to minimize $P(FP)$ by restricting the value of $P(X)$ below a specific threshold. The threshold represents the data usage limit of the backup path and should be adaptable to different specified budgets. Thus, we limit the backup path utilization rate to be less than or equal to the frame stall rate multiplied by a parameter δ:

$$P(X) = \sum_{S_i \in X} P(S_i) \leq \sum_{S_i \in U} P(stall \mid S_i) P(S_i) \cdot \delta \quad (6)$$

With the maximization target (Eqn. 4), the constraint (Eqn. 6), and the probability model M, we can formulate the backup path utilization strategy derivation as an Integer Linear Programming problem:

$$\begin{aligned} &\text{Maximize} \sum_{S_i \in U} P(stall \mid S_i) P(S_i) P_{backup}(S_i) \cdot x_i \\ &x_i = 0, 1 \\ &\text{Subject to} \sum_{S_i \in U} P(S_i) \cdot x_i \leq \sum_{S_i \in U} P(stall \mid S_i) P(S_i) \cdot \delta \end{aligned} \quad (7)$$

In Eqn. 7, the maximization term aims to achieve objective **T1**, which is to cover delayed frames, and the constraint term aims to achieve objective **T2**, which is to minimize the data usage of the backup path. By solving the ILP problem, the sender can obtain a strategy X and determine when to use the backup path.

3.5 State Manager

The state manager continuously monitors the capacity of the Wi-Fi and cellular path for each user session. It creates and manages two state probability models for the multipath agent, one for the decision on frame retransmission and the other for the primary path switch. Nonetheless, we encounter two challenges in practice: *i)* What information should be extracted from sender statistics to form a state space; *ii)* While the overall frame stall rate is assumed to be a constant value, how can we obtain the distribution of $P(stall \mid S_i)$ and $P(S_i)$.

3.5.1 *State space formulation.*

In principle, it is possible to extract all available information from sender statistics to form a high-dimensional state space. However, we take two factors into consideration when deciding on the state space formulation: *i)* A multi-dimensional state space increases the difficulty and complexity of maintaining a probability model and deriving a backup path utilization strategy based on it; *ii)* Some information, such as RTT and acknowledged bytes, is already utilized by CCAs to maintain low median latency, making it unnecessary to include them in the state space. Therefore, we choose to use a single-dimension state space for both probability models.

Frame retransmission state space. We utilize the in-flight time of all frames to create a state space U_1 for deciding when to retransmit frames on the cellular path. Specifically, we calculate the in-flight time of a video frame once it is acknowledged, and this value is used to represent a point in the state space. We choose to use the per-frame in-flight time as the input state because it allows us to derive a probability model that can indicate the correlation between the stall rate and the frame in-flight time. This, in turn, provides us with the necessary information to promptly retransmit frames on the backup path.

Primary path switch state space. To derive the state probability model for the primary path switch decision, we use the in-flight time of the earliest unacknowledged frame to form a state space U_2. This choice reflects the frame delivery delay

of the primary path, providing clues on the degradation of the primary path's capacity and the chance for a switch. In practice, the state manager periodically inspects the in-flight time of all unacknowledged frames and sets the in-flight time of the earliest dispatched frame as the current state.

Cellular path capacity monitoring. Since AUGUR uses the cellular as the backup path, which can also be susceptible to fluctuations [39, 40, 41] and lead to prolonged latency, the state manager needs to monitor the characteristics of the cellular path to estimate its capacity and determine whether it is sufficient to perform frame transmission. As we use the in-flight time of frames as input states, at state S_i, the frame has been sent for S_i time. Therefore, to perform effective retransmission or primary path switch to reduce long tail latency, the RTT of the cellular path should not exceed $T_{thresh} - S_i$. Since a frame delivery latency greater than 200 ms is highly likely to cause a video stall [8], we set T_{thresh} to be 200 ms. Hence, we evaluate the RTT as a metric of the backup path capacity using the following equation:

$$P_{backup}(S_i) = P(\text{RTT}_{cell} \leq T_{thresh} - S_i) \quad (8)$$

However, we cannot obtain $P(\text{RTT}_{cell} \leq T_{thresh} - S_i)$ passively from receiver feedback because the cellular path acts as a backup, and the feedback is intermittent. Therefore, the state manager actively probes the cellular path by periodically sending an 8-byte PING message (with an interval of 50 ms) to monitor the RTT. Since the backup cellular path is infrequently used and other applications using cellular interfaces (*e.g.,* voice call, video conferencing) are typically inactive during cloud gaming sessions, small-sized packets are efficient for probing the path RTT. The overhead of the probe messages is less than 0.06% in theory, compared to the high frame rate at which large-sized video frames (typically over 4KB) are transmitted (*e.g.,* 60 fps).

3.5.2 Probability distribution derivation

The state probability model M contains the distributions of $P(stall \mid S_i)$ and $P(S_i)$. However, they are difficult to accurately describe. Firstly, these distributions could be arbitrary over time for different users. Secondly, the state manager cannot predict future states or stalls, and can only deduce an approximate distribution based on recorded states. To address this challenge, we leverage our observation discussed in §2.4 that the user characteristics remain stable for an individual user over a period of time. Based on this, we assume that the probability distributions of states remain fixed for a time window. Within a time window Δ, we can obtain the frequency of each state S_i and the corresponding stall event, denoted as $\hat{P}(S_i)$ and $\hat{P}(stall \mid S_i)$. We use these frequencies to approximate the probability distribution of the states in this time window. The approximated stall rate $\hat{P}(stall)$ can be derived by Eqn. 1.

In this way, the state manager continuously updates the states explained in §3.5.1. For each time window Δ, it creates and maintains two approximated state probability models $\hat{M}_{1,2} = \{\hat{P}(stall), \hat{P}(stall \mid S_i), \hat{P}(S_i)\}, S_i \in U_{1,2}$ with two different state spaces. The value of Δ reflects the level of fluctuation in a single user's characteristics, and its effect on AUGUR is evaluated in §5.4.

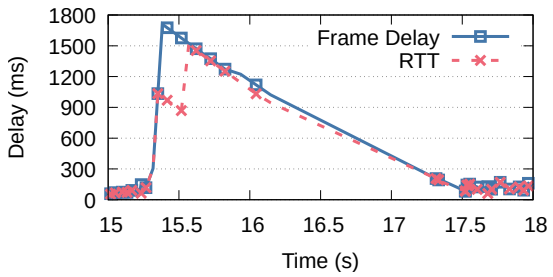


Figure 11: Wi-Fi capacity degradation causes bursty frame stalls lasting for hundreds of milliseconds.

3.6 Multipath Agent

The multipath agent plays a critical role in deciding whether to retransmit in-flight frames on the cellular path and whether to switch the primary path from Wi-Fi to cellular. To make these decisions, it utilizes the state probability models $\hat{M}_1$ and $\hat{M}_2$ provided by the state manager and plugs them into the ILP problem described in Eqn. 7. It is typically time-consuming and computation-intensive to solve an ILP problem. Nevertheless, when considering our single-dimensional state spaces $U_{1,2}$, and utilizing RTT-based backup path capacity specified in Eqn. 8, instead of looping each combination of states, we only need to determine a threshold in the continuous state space to find an optimal strategy. Therefore, the ILP problem in Eqn. 7 transforms into a linear-time solution, facilitating quick resolution. Based on the output of the simplified ILP problems, the multipath agent obtains the necessary decisions to reduce long tail latency and minimize cellular cost.

3.6.1 Application Level Frame Retransmission

Existing multipath schedulers usually only decide which path to use for delivering incoming new data at packet departure time. As presented in §2.2, unpredictable RTT inflation of wireless links could severely delay the scheduled packets. Therefore, to reduce long tail latency, it is important to track the in-flight data and actively retransmit it when it is at risk of being delayed.

In order to eliminate as much tail latency as possible, AUGUR introduces application-level frame retransmission by promptly retransmitting in-flight frames on the cellular path to prevent delays. We choose to retransmit the entire frame instead of individual data packets because: *i)* An RTT inflation at the wireless last hop would typically result in delays for the majority of packets within a frame (*e.g.,* we observe that when the frame delivery delay surpasses 200 ms, the delay of the initial packet in the frame exceeds 200 ms in 71.8% of cases), making it reasonable to actively retransmit the

entire frame through an alternative path; *ii)* Implementing frame retransmission in userspace is straightforward, rendering it a portable and easily deployable solution across a wide range of scenarios.

Naturally, there is a strong relationship between the in-flight time of an unacknowledged frame and the possibility the frame would be severely delayed, and such a relationship is quantified by our state probability model $\hat{M}_1$, which uses frame in-flight time as its state space. Consequently, as shown in Fig. 10, while the state manager keeps updating the probability model $\hat{M}_1$ during the streaming process, the multipath agent continuously solves the simplified ILP problem using $\hat{M}_1$ as input and obtains a decision X_1. It constantly monitors all unacknowledged frames and retransmits those whose in-flight time T matches with a state in X_1 (*i.e.,* $T = S_i \in X_1$). By doing so, AUGUR can proactively rescue frames that are at risk of stalling while also adhering to limited usage of the backup path. We evaluate the effectiveness of application-level frame retransmission in §5.4.

3.6.2 Primary Path Switch Scheduling

Unlike existing multipath schedulers that schedule paths for each data packet, AUGUR chooses the Wi-Fi path as the primary path and schedules frames on it by default, while the cellular path should only be when necessary to rescue delayed frames. However, in practice, we observe that severe Wi-Fi capacity degradation can occur occasionally. As demonstrated in Fig. 11, such degradation results in bursty frame stalls that last for hundreds of milliseconds, which can have a significant impact on user experience.

To address this issue, the multipath agent can temporarily switch the primary path to cellular and stop sending new frames on the degraded Wi-Fi path. Similar to the frame retransmission decision procedure, the multipath agent continuously solves the simplified ILP problem using $\hat{M}_2$ provided by the state manager and obtains a decision X_2. It initiates a primary path switch when two conditions are met: *i)* the in-flight time of the earliest unacknowledged frame T_1 matches with a state in X_2 (*i.e.,* $T_1 = S_i \in X_2$) and *ii)* the RTT of the cellular path is lower than that of the Wi-Fi path. In this scenario, AUGUR sends all newly generated frames through the cellular path and stops deriving backup path utilization strategies. In addition, the multipath agent promptly retransmits all in-flight frames of the Wi-Fi path through the cellular path since they are likely to experience prolonged delays upon detection of capacity degradation. However, directly using the cellular path as the primary path violates our cellular data limit rule, and since the capacity degradations of the Wi-Fi path are transient, it is necessary to switch back to Wi-Fi as soon as possible. While new frames are being sent through the cellular path, the sending queue of the Wi-Fi path can be drained, and we start sending probe frames on the Wi-Fi path when there are fewer than two in-flight frames on the Wi-Fi path to detect any capacity improvements. Once the capacity of the Wi-Fi path recovers (*i.e.,* the Wi-Fi path RTT is not in X_2), the multipath agent switches the primary path back to Wi-Fi and restores the simplified ILP problem-solving process. The effectiveness of primary path transition is evaluated in §5.4.

4 Trace-driven Emulation

In this section, we evaluate AUGUR in an emulation environment with real-world traces to compare it with other multipath transport schemes. Based on the results of our emulation, we further conduct large-scale experiments on our cloud gaming platform (§5).

4.1 Evaluation Methodology

Testbed. We use *mpshell* [42], a multipath extension of Mahimahi [43] for network emulation. We develop a testbed framework to implement the streaming pipeline in Fig. 1 with approximately 6000 lines of Python code. To demonstrate the cooperation between AUGUR and CCAs, we also implement two CCAs designed for real-time streaming including Salsify [11] and SQP [12].

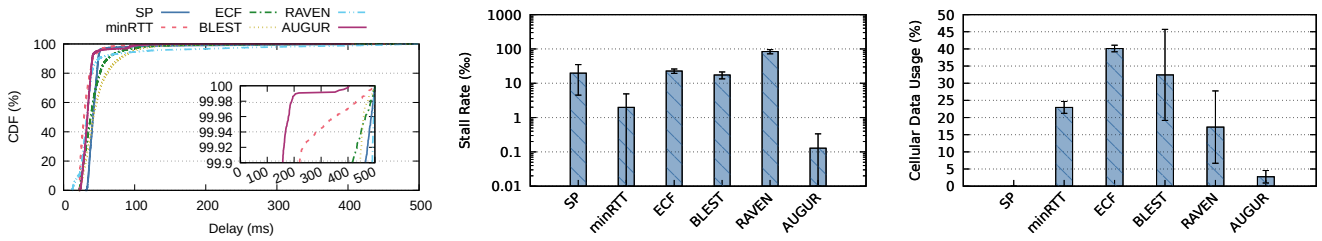
Trace collection. We collect Wi-Fi and cellular link traces based on the running logs of our user sessions. The logs contain the *user-perceived* wireless network bandwidth, and RTT inflation caused by wireless fluctuation would lead to a sudden decrease in user-perceived bandwidth. To evaluate the effectiveness of AUGUR in the presence of wireless fluctuations, we filter out traces that were either too brief (less than 10 minutes) or too consistent (with no RTT inflation). Each trace is replayed for more than 10 minutes.

Baseline. We compare AUGUR with the following multipath transport schemes as baselines:

- Single Path (SP): all video frames are streamed exclusively through the Wi-Fi path;
- minRTT [16]: the default multipath scheduler of MPTCP, which schedules packets through the path with the lowest estimated RTT;
- ECF [19]: it utilizes some relevant information (*e.g.,* `cwnd` value) of a path besides RTT to provide available aggregate bandwidth of all paths. It assumes that the underlying CCAs are congestion-window-based and require `cwnd` values for decision-making. For the rate-based CCAs we use, we derive `cwnd` value by multiplying the sending rate and the estimated RTT.
- BLEST [18]: it aims to avoid HoL-blocking and spurious retransmissions by controlling the buffer blocking. We provide it with the `cwnd` value in the same way as described above.
- RAVEN [17]: it replicates packets on multiple paths when confidence about network latency predictions is low to reduce latency.

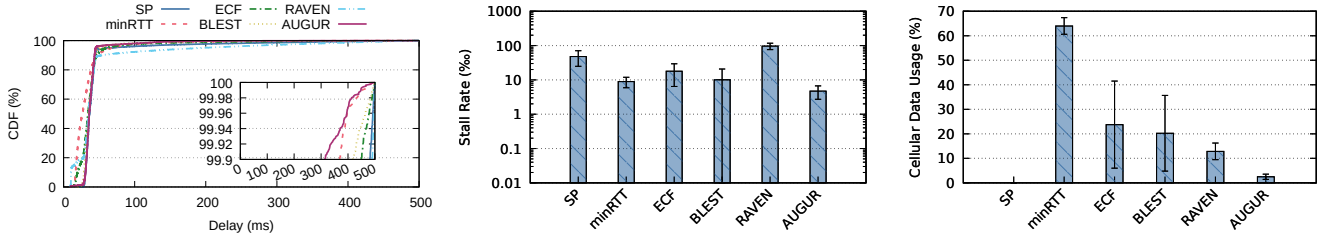
4.2 Performance

Frame delivery delay and stall rate. We measured the frame delivery delay for all multipath transport schemes with



(a) Frame delivery delay. (b) Frame stall rate. (c) Cellular data usage.

Figure 12: AUGUR performance cooperating with Salsify.



(a) Frame delivery delay. (b) Frame stall rate. (c) Cellular data usage.

Figure 13: AUGUR performance cooperating with SQP.

different CCAs. As shown in Fig. 12a and Fig. 13a, while the median frame delivery delays are similar for all multipath schemes, AUGUR effectively reduces tail latency and outperforms other multipath schemes for both CCAs. Compared to only using Wi-Fi for frame transmission, AUGUR reduces the 99.9th percentile latency by 66.0% and 35.4% for Salsify and SQP, respectively. As a result, AUGUR effectively reduces the frame stall rate, as shown in Fig. 12b and Fig. 13b. Among all multipath schemes, AUGUR achieves the lowest average frame stall rate, and compared to using only the Wi-Fi path, AUGUR reduces the stall rate by 99.5% and 91.5% for Salsify and SQP, respectively. This demonstrates that AUGUR cooperates with different CCAs and effectively reduces long tail latency in real-time streaming.

Cellular data usage. To evaluate AUGUR's ability to reduce cellular data usage, we calculate the ratio of bytes transmitted on the cellular path to the total transmitted bytes for all multipath schemes. Our results, as shown in Fig. 12c and Fig. 13c, indicate that AUGUR achieves the lowest cellular data usage among all multipath schemes, with only 2.7% and 2.3% cellular data usage for Salsify and SQP, respectively. It is worth noting that compared to minRTT, which can also effectively reduce the frame stall rate, AUGUR reduces cellular data usage by 88.1% and 96.4% for Salsify and SQP, respectively. These findings demonstrate that AUGUR can effectively reduce cellular data usage without compromising performance, making it a viable option for deployment in commercial platforms.

Multipath scheme	SP	minRTT	AUGUR
Num. of user sessions	3974	4167	3699
Total		11840	

Table 1: Large-scale experiment setup.

5 Large-scale Deployment in the Wild

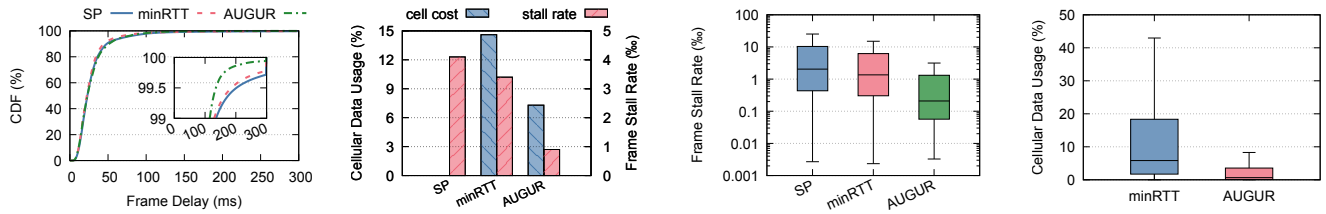
We further deploy AUGUR in Tencent Start cloud gaming server clusters to evaluate its effectiveness in reducing long tail latency and constraining cellular data usage.

5.1 Implementation and Deployment

We implement AUGUR in our cloud gaming platform fully in the userspace. The streaming application on user devices initiates separate connections to our servers with two sockets to leverage the Wi-Fi and the cellular path. Regarding the cellular budget, users are required to set a per-month limit in MB, and AUGUR calculates the corresponding value of δ based on the average stall rate and playing time of all users. For instance, a limit of 500 MB per month would result in a δ value of 3.5. We set the default value for the time window Δ to be 5 min. We have deployed AUGUR for over six months and served millions of users with over ten million hours of playing time.

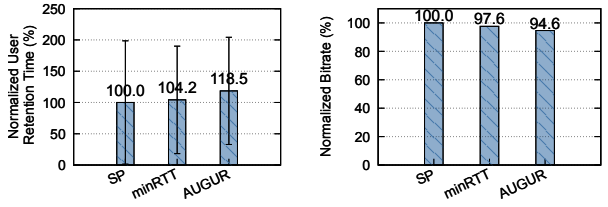
5.2 Experiment Setup

We conduct large-scale A/B tests to evaluate AUGUR in our platform. In addition to AUGUR and single-path Wi-Fi (SP), we also deployed minRTT [16] based on our emulation results in §4.2. We initiate SP, minRTT, and AUGUR for all users who are willing to use cellular interfaces for better performance, and each user is served by each with equal possi-



(a) Frame delivery delay. (b) Average frame stall rate and cellular data usage. (c) Frame stall rate. (d) Cellular data usage.

Figure 14: Online performance of multipath transport schemes.



(a) User retention time. (b) Average bitrate.

Figure 15: Online user QoE of multipath transport schemes.

bility. All other settings and implementations, such as CCA, remain the same. The A/B tests are conducted for two weeks and result in a total of 11840 user sessions. The numbers of user sessions for each scheme are shown in Tab. 1.

5.3 Performance

Frame delivery delay and stall rate. Achieving consistent low latency is a key requirement for real-time services that require interactivity. To evaluate the performance of frame transport delay in AUGUR, we plot the CDFs of frame delivery delays and frame stall rate for all user sessions using three multipath schemes in Fig. 14a, Fig. 14b, and Fig. 14c. While the median delay is similar across all schemes, AUGUR outperforms SP and minRTT in reducing long tail delay at the 99th percentile, achieving reductions of 14.2% and 7.7% respectively. More importantly, for severely prolonged tail delays (*i.e.,* frame delay $\geq$ 200 ms) that significantly impact user experience, AUGUR is capable of maintaining the 99.9th percentile latency below 200 ms, while SP and minRTT leave the 99.9th percentile latency over 300 ms.

Consequently, AUGUR reduces the average stall rate of all user sessions to below 0.1%, improving by 78.0% and 73.5% compared to SP and minRTT, respectively. In addition, AUGUR reduces the median stall rate of all user sessions to only 0.02%, and reduces the median stall rate by 90.0% and 85.7% compared to SP and minRTT, respectively. These results demonstrate that in the presence of RTT inflation caused by wireless fluctuation, AUGUR is able to reduce the long tail latency and frame stall rate to enhance the service experience for the majority of mobile users.

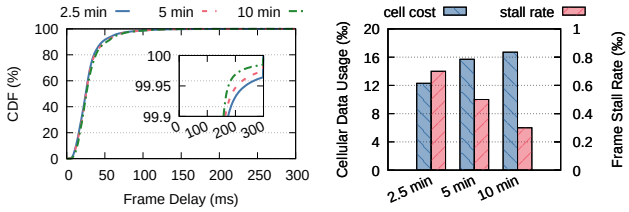
Cellular cost. In order to compare the cellular data usage of AUGUR and the baseline minRTT multipath scheduler, we calculate the ratio of bytes transmitted on the cellular path to the total transmitted bytes for each user session. As shown in Fig. 14b and Fig. 14d, on average, AUGUR sends 7.3% of the video data streaming through the cellular path, which is 2$\times$ less than minRTT. Additionally, for all user sessions, AUGUR incurs 0.65% median cellular data usage, which is 8.9$\times$ less than minRTT. This reduction in cellular data usage can be attributed to two factors. First, AUGUR retransmits only those frames that are likely to be delayed, instead of scheduling all frames on the path with a lower RTT. Second, AUGUR strictly constrains the cellular budget according to Eqn. 6, resulting in a controlled cellular usage pattern. Note that this constraint allows AUGUR to bound the maximum cellular cost and reduce it by 4.7$\times$ compared to minRTT scheduler.

User QoE. We use user retention time and average video bitrate as our QoE metrics. As discussed in §2.1, a higher frame stall rate leads to a decrease in user retention time, which represents a degradation in the user experience of our service. To illustrate the effect of AUGUR on user experience, we show the normalized average user retention time of three transport schemes in Fig. 15a, which demonstrates that with AUGUR deployed, user retention time can be improved by 18.5% and 13.7% on average compared to SP and minRTT, respectively. Note that user retention time has a large deviation because it is also affected by other factors that are uncorrelated to network performance, such as personal preference.

As AUGUR actively retransmits frames on other paths and switches the primary path, it may interfere with the bandwidth estimation mechanism of the underlying transport (*e.g.,* CCA), thus affects the user QoE. We show the normalized average video bitrate of three transport schemes in Fig. 15b, which demonstrates that AUGUR has little impact on video bitrate, with 5.4% and 3.1% decrease compared to SP and minRTT, respectively.

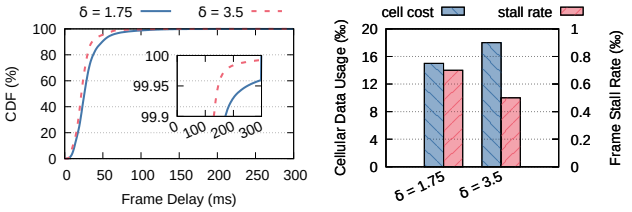
5.4 Micro-benchmark

We further perform some micro-benchmarks to investigate the impact of different design choices made in the development of AUGUR, including parameter settings and cellular utilization approaches.



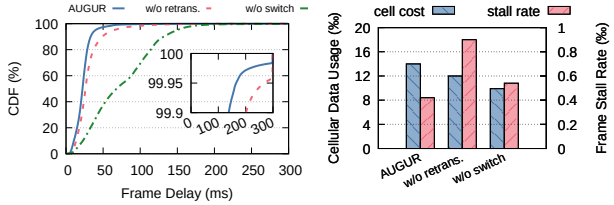
(a) Frame delivery delay. (b) Stall rate and cellular cost.

Figure 16: Impact of time window value Δ.



(a) Frame delivery delay. (b) Stall rate and cellular cost.

Figure 17: Impact of cellular constraint factor δ.



(a) Frame delivery delay. (b) Stall rate and cellular cost.

Figure 18: Impact of frame retransmission and primary path switch.

5.4.1 Parameter Setting

Time window value. AUGUR uses a time window Δ to approximate the state probability models. The value of Δ reflects the fluctuation level of a single user's characteristics. To investigate the impact of Δ, we conduct an experiment involving 2,186 users, with Δ set to 2.5 min, 5 min, and 10 min. As illustrated in Fig. 16, we observe that AUGUR is not sensitive to the choice of Δ, and the frame delivery delays remain similar across all three settings. Additionally, a larger time window results in slightly higher cellular costs and a lower frame stall rate. This is because a larger time window can detect more stall events and adopt a more aggressive backup path utilization strategy. In practice, we set Δ to 5 min based on our deployment experience to reach a balance between stall rate and cellular cost.

Cellular constraint factor. AUGUR uses a factor δ to constrain the cellular cost in Eqn. 6 and its value adapts to the cellular budget specified by the user. To demonstrate the effect of different δ values, we show the performance of AUGUR with two most frequently specified settings: $\delta = 1.75$ corresponding to 250 MB per month and $\delta = 3.5$ corresponding to 500 MB per month. As shown in Fig. 17, a larger constraint factor results in a more effective reduction of long tail latency and stall rate with higher cellular data usage. This demonstrates that δ controls the trade-off between better network performance and lower cellular data cost. We leave the decision of choosing δ to each user based on their data budget.

5.4.2 Cellular Path Utilization

AUGUR utilizes the cellular path in two approaches: frame retransmission and primary path switch. To investigate the effectiveness of both approaches, we conduct an experiment with 7,266 invited users accessing our real-time streaming platform. We randomly assign one-third of users with frame retransmission disabled, one-third of users with primary path switch disabled, and the rest with full AUGUR functionality.

Frame retransmission. AUGUR uses application-level frame retransmission to rescue delayed frames on the Wi-Fi path to reduce long tail latency. As shown in Fig. 18, compared to only performing the primary path switch, enabling frame retransmission reduced the 99th percentile latency and stall rate by 45.8% and 55.5%, respectively, with 0.2% higher cellular data usage. This demonstrates that leveraging the cellular path to perform application-level frame retransmission is necessary to reduce long tail latency caused by RTT inflation.

Primary path switch. To cope with bursty frame stalls caused by severe Wi-Fi capacity degradation, AUGUR schedules the primary path to cellular. As depicted in Fig. 18, the decision to switch the primary path avoids streaming frames on a degraded path and can significantly reduce long tail latency. Compared to only retransmitting frames, AUGUR reduces the 99th percentile latency and stall rate by 25% and 31.7%, respectively, with only a 0.4% increase in cellular cost.

6 Discussion

In this section, we discuss some potential limitations and future work of AUGUR.

Deployment scenario. As a cloud gaming service provider, we primarily evaluated the performance of AUGUR in our cloud gaming clusters and demonstrated its ability to reduce long tail latency. Meanwhile, the scenario of using both Wi-Fi and cellular paths for real-time streaming can be applied to many other applications with consistently low latency requirements, such as video conferencing and live streaming. With the development of Wi-Fi 7 [44] and 5G [41], which provide higher network bandwidth, AUGUR can also be deployed for AR/VR applications across a wide range.

Multiple cellular paths. AUGUR currently leverages only one cellular path. However, as multi-carrier phones become increasingly popular and the benefits of a multipath system design that supports more than two paths are being acknowledged [45], AUGUR can be extended to support multiple cel-

lular paths. This would allow AUGUR to enable multiple cellular backup paths and constrain their usage by introducing different δ values for each path. We leave the extension of AUGUR to support multiple cellular paths as our future work.

Cross-layer optimization. Recent years have seen the proposal of various approaches to improve the performance and user QoE of mobile real-time streaming. These approaches include deploying dedicated CCAs [9, 11, 12], using the adaptive bitrate (ABR) to meet network conditions [46, 47, 48, 49], leveraging forward error correction (FEC) to recover lost frames [50, 51, 28], encoding video frames in a QoE-aware manner [52, 53, 54], allocating and scheduling streaming resources [55, 56, 57], utilizing timely feedback from Wi-Fi APs [58], and adjusting frame rate based on network conditions [8, 59]. As a multipath transport service, AUGUR is orthogonal to these approaches and can be used jointly with them. Furthermore, AUGUR can be extended to perform cross-layer optimization with these approaches (*e.g.,* retransmit frames with FEC). We leave cross-layer optimization of AUGUR as our future work.

7 Related Work

Multipath transport. Multipath transport utilizes multiple paths simultaneously for data delivery in a single connection. MPTCP [60] and MPUDP [61] integrate multipath transport into the OS kernel and provide a single-connection abstraction to applications. Based on this framework, many multipath schedulers have been proposed, such as ECF [19], BLEST [18], RAVEN [17], and DEMS [20]. However, the large-scale deployment of such a framework has been slow due to its modification of user device kernels [16] and network middleboxes [62]. Therefore, some multipath transport systems based on QUIC have been proposed, such as MPQUIC [63], PQUIC [64], and XLINK [24]. Additionally, there are some multipath transport schemes using TCP splitting (POLYCORN [65]) or implemented in the application layer (MP-H2 [66], MSPlayer [67], MP-DASH [21], mHTTP [68]). Like these approaches, AUGUR can be deployed on a large scale and is specifically designed for real-time streaming with low tail latency requirements.

Cellular cost reduction. Several approaches have been proposed to reduce the cellular data cost for network traffics. By leveraging the delay-tolerant nature of VoD applications, MP-DASH [21] schedules the cellular path based on a user-specified preference and a known video chunk size, and Obilgir *et al.* schedules data transmissions to reduce cellular data usage. TrafficGuard [69] adopts a proxy-based method to reduce cellular traffic using a network-layer VPN. Yanyuan *et al.* [70] uses a chunk filter and a rate adaptation algorithm to reduce cellular usage in video streaming. In contrast to all approaches above, AUGUR is designed for real-time streaming applications and reduces cellular cost by directly avoiding unnecessary data transmission.

8 Conclusion

In this paper, we demonstrate that long tail latency can severely degrade the user experience in real-time streaming. We further find that in wireless networks like Wi-Fi, long tail latency is induced by RTT inflation caused by wireless fluctuation, and thus cannot be reduced by deploying dedicated CCAs. While it is straightforward to leverage cellular paths to alleviate the impact of Wi-Fi RTT inflation, we reveal that users express strong concern about cellular cost, thus it is crucial to minimize cellular data usage for a widely deployed multipath transport service. We design AUGUR, a practical multipath transport service for mobile real-time streaming that meets the requirement of reducing long tail latency, minimizing cellular data usage, and enabling large-scale deployment. We deploy AUGUR on our cloud gaming platform with millions of users across a wide area and demonstrate its effectiveness in reducing long tail latency and frame stall rate with minimal cellular data usage. We believe that AUGUR can be widely deployed at any scale to provide universal low-latency mobile streaming access.

Acknowledgements

We are grateful to the NSDI reviewers for their constructive critique, and to our shepherd Akshay Narayan in particular, for his valuable comments, all of which have helped us greatly improve this paper. This work is supported by National Key Research and Development Plan, China (Grant No. 2020YFB1710900), National Natural Science Foundation of China (Grant No. 62022005 and 62172008).

References

[1] Samsung gaming hub. https://www.samsung.com/us/televisions-home-theater/tvs/gaming-hub/, 2022.

[2] Google cloud gaming. https://cloud.google.com/solutions/games, 2023.

[3] Zoom: One platform to connect. https://zoom.us, 2023.

[4] Microsoft teams. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software, 2023.

[5] Google map live view support. https://support.google.com/maps/answer/9332056?hl=en&co=GENIE.Platform%3DiOS, 2023.

[6] Youtube vr - home. https://vr.youtube.com/, 2023.

[7] Cloud gaming market: Global industry trends, share, size, growth, opportunity and forecast 2023-2028. Market report 5732901, IMARC Group, 2023.

[8] Zili Meng, Tingfeng Wang, Yixin Shen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. Enabling high quality Real-Time communications with adaptive Frame-Rate. In *USENIX NSDI*, 2023.

[9] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *ACM MMSys*, 2016.

[10] Xiaoqing Zhu and Rong Pan. Nada: A unified congestion control scheme for low-latency interactive video. In *20th IEEE International Packet Video Workshop*, 2013.

[11] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *USENIX NSDI*, 2018.

[12] Devdeep Ray, Connor Smith, Teng Wei, David Chu, and Srinivasan Seshan. Sqp: Congestion control for low-latency interactive video streaming. *arXiv preprint arXiv:2207.11857*, 2022.

[13] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM*, 2015.

[14] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *USENIX NSDI*, April 2013.

[15] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. Experimental evaluation of multipath tcp schedulers. In *ACM CSWS*, 2014.

[16] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath tcp. In *USENIX NSDI*, 2012.

[17] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. Raven: Improving interactive latency for the connected car. In *ACM MobiCom*, 2018.

[18] Per Hurtig, Karl-Johan Grinnemo, Anna Brunstrom, Simone Ferlin, Özgü Alay, and Nicolas Kuhn. Low-latency scheduling in mptcp. *IEEE/ACM Transactions on Networking*, 2019.

[19] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *ACM CoNEXT*, 2017.

[20] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. Accelerating multipath transport through balanced subflow completion. In *ACM MobiCom*, 2017.

[21] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In *ACM CoNEXT*, 2016.

[22] Cheng-Lin Tsao and Raghupathy Sivakumar. On effectively exploiting multiple wireless interfaces in mobile hosts. In *ACM CoNEXT*, 2009.

[23] Onjeinika Brooks. Best cell phone plans of 2022. https://www.reviews.org/internet-service/best-internet-service-providers/, 2022.

[24] 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. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *ACM SIGCOMM*, 2021.

[25] Start cloud gaming. https://start.qq.com/, 2023.

[26] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 2003.

[27] Taehyun Kim, Niranjan Avadhanam, and Sridharan Subramanian. Dimensioning receiver buffer requirement for unidirectional vbr video streaming over tcp. In *IEEE International Conference on Image Processing*, 2006.

[28] Michael Rudow, Francis Y. Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and K.V. Rashmi. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *USENIX NSDI*, 2023.

[29] Serhat Arslan, Yuliang Li, Gautam Kumar, and Nandita Dukkipati. Bolt: Sub-RTT congestion control for Ultra-Low latency. In *USENIX NSDI*, 2023.

[30] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. Hpcc: High precision congestion control. In *ACM SIGCOMM*, 2019.

[31] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. Pint: Probabilistic in-band network telemetry. In *ACM SIGCOMM*, 2020.

[32] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *ACM SIGCOMM Computer Communication Review*, 2002.

[33] Alexander Frommgen, Tobias Erbshäußer, Alejandro Buchmann, Torsten Zimmermann, and Klaus Wehrle. Remp tcp: Low latency multipath tcp. In *IEEE ICC*, 2016.

[34] Markus Amend, Veselin Rakocevic, and Joachim Habermann. Cost optimized multipath scheduling in 5g for video-on-demand traffic. In *IEEE WCNC*, 2021.

[35] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 1951.

[36] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 1951.

[37] Xinlei Yang, Hao Lin, Zhenhua Li, Feng Qian, Xingyao Li, Zhiming He, Xudong Wu, Xianlong Wang, Yunhao Liu, Zhi Liao, Daqiang Hu, and Tianyin Xu. Mobile access bandwidth in practice: Measurement, analysis, and implications. In *ACM SIGCOMM*, 2022.

[38] Haiqing Jiang, Zeyu Liu, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Understanding bufferbloat in cellular networks. In *ACM SIGCOMM CellNet*, 2012.

[39] Jing Wang, Yufan Zheng, Yunzhe Ni, Chenren Xu, Feng Qian, Wangyang Li, Wantong Jiang, Yihua Cheng, Zhuo Cheng, Yuanjie Li, Xiufeng Xie, Yi Sun, and Zhongfeng Wang. An active-passive measurement study of tcp performance over lte on high-speed rails. In *ACM MobiCom*, 2019.

[40] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *ACM MobiSys*, 2012.

[41] 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. A variegated look at 5g in the wild: Performance, power, and qoe implications. In *ACM SIGCOMM*, 2021.

[42] R. Netravali A. Sivaraman and K. J. Winstein. Mpshell. https://github.com/ravinet/mahimahi/releases/tag/old%2Fmpshell_scripted, 2020.

[43] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate record-and-replay for http. In *USENIX ATC*, 2015.

[44] Cailian Deng, Xuming Fang, Xiao Han, Xianbin Wang, Li Yan, Rong He, Yan Long, and Yuchen Guo. Ieee 802.11be wi-fi 7: New challenges and opportunities. *IEEE Communications Surveys & Tutorials*, 2020.

[45] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y. Ethan Guo, Feng Qian, and Z. Morley Mao. Mpbond: Efficient network-level collaboration among personal mobile devices. In *ACM MobiCom*, 2020.

[46] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *ACM SIGCOMM*, 2017.

[47] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: Auto-tuning video abr algorithms to network conditions. In *ACM SIGCOMM*, 2018.

[48] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *ACM SIGCOMM*, 2020.

[49] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. Gso-simulcast: Global stream orchestration in simulcast video conferencing systems. In *ACM SIGCOMM*, 2022.

[50] C. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 1998.

[51] Justin Uberti. Webrtc forward error correction requirements. *draft-ietf-rtcweb-fec-01 (work in progress)*, 2015.

[52] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. Pano: Optimizing 360° video streaming with a better understanding of quality perception. In *ACM SIGCOMM*, 2019.

[53] Devdeep Ray, Jack Kosaian, K. V. Rashmi, and Srinivasan Seshan. Vantage: Optimizing video upload for time-shifted viewing of social live streams. In *ACM SIGCOMM*, 2019.

[54] Mallesham Dasari, Kumara Kahatapitiya, Samir R. Das, Aruna Balasubramanian, and Dimitris Samaras. Swift: Adaptive video streaming with layered neural codecs. In *USENIX NSDI*, 2022.

[55] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, Chen Sun, Gareth Tyson, and Hongqiang Harry Liu. Livenet: A low-latency video transport network for large-scale live streaming. In *ACM SIGCOMM*, 2022.

[56] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *ACM MobiCom*, 2015.

[57] Le Xu, Shivaram Venkataraman, Indranil Gupta, Luo Mai, and Rahul Potharaju. Move fast and meet deadlines: Fine-grained real-time stream processing with cameo. In *USENIX NSDI*, 2021.

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

[59] Tingfeng Wang, Zili Meng, Mingwei Xu, Rui Han, and Honghao Liu. Enabling high frame-rate uhd real-time communication with frame-skipping. In *ACM HotEdgeVideo*, 2021.

[60] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. Tcp extensions for multipath operation with multiple addresses. Technical report, 2013.

[61] Daniel Lukaszewski and Geoffrey Xie. Multipath transport for virtual private networks. In *10th USENIX Workshop on Cyber Security Experimentation and Test*, 2017.

[62] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z. Morley Mao, and Subhabrata Sen. An in-depth understanding of multipath tcp on mobile devices: Measurement and system design. In *ACM MobiCom*, 2016.

[63] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *ACM CoNEXT*, 2017.

[64] Quentin De Coninck, François Michel, Maxime Piraux, Florentin Rochet, Thomas Given-Wilson, Axel Legay, Olivier Pereira, and Olivier Bonaventure. Pluginizing quic. In *ACM SIGCOMM*, 2019.

[65] Yunzhe Ni, Feng Qian, Taide Liu, Yihua Cheng, Zhiyao Ma, Jing Wang, Zhongfeng Wang, Gang Huang, Xuanzhe Liu, and Chenren Xu. POLYCORN: Data-driven cross-layer multipath networking for high-speed railway through composable schedulerlets. In *USENIX NSDI*, 2023.

[66] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z. Morley Mao. Mp-h2: A client-only multipath solution for http/2. In *ACM MobiCom*, 2019.

[67] Yung-Chih Chen, Don Towsley, and Ramin Khalili. Msplayer: Multi-source and multi-path video streaming. *IEEE Journal on Selected Areas in Communications*, 2016.

[68] Juhoon Kim, Yung-Chih Chen, Ramin Khalili, Don Towsley, and Anja Feldmann. Multi-source multipath http (mhttp): A proposal. In *ACM SIGMETRICS*, 2014.

[69] Zhenhua Li, Weiwei Wang, Tianyin Xu, Xin Zhong, Xiang-Yang Li, Yunhao Liu, Christo Wilson, and Ben Y. Zhao. Exploring Cross-Application cellular traffic optimization with baidu TrafficGuard. In *USENIX NSDI*, 2016.

[70] Yanyuan Qin, Shuai Hao, Krishna R. Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. Quality-aware strategies for optimizing abr video streaming qoe and reducing data usage. In *ACM MMSys*, 2019.