TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery

Haiping Wang
ByteDance
wanghaiping.paloma@bytedance.com

Zhenhua Yu*
Meituan
yuzhenhua02@meituan.com

Ruixiao Zhang
The University of Hong Kong
zrxhku@hku.hk

Siping Tao
ByteDance
siping.tao@bytedance.com

Hebin Yu
ByteDance
yuhebin.824@bytedance.com

Shu Shi
ByteDance
shishu.1513@bytedance.com

ABSTRACT

Ultra-low latency video streaming has received explosive growth in the past few years. However, existing methods all focus on single-path transmission, which is ineffective in dealing with really poor network conditions. To tackle their problems, we propose TwinStar, a novel multi-path framework to improve the experience quality of ultra-low latency video. The core idea of TwinStar is to concurrently leverage multiple paths to mitigate the negative impacts of network jitter on a single path. In particular, by carefully designing the video encoding, data allocation and loss recovery, TwinStar is very robust to handle network dynamics and deliver high-quality video services. We have deployed TwinStar in a commercial cloud gaming platform and evaluated it with real-world networks. The extensive experiments demonstrate that TwinStar significantly outperforms the single-path transmission methods, with 91% reduction in stall ratio and 11% improvement in PSNR across all regions.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

Cloud Gaming, Multi-path, Video Delivery

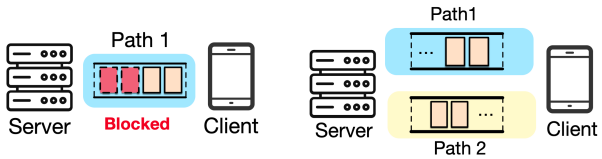
ACM Reference Format:

Haiping Wang, Zhenhua Yu, Ruixiao Zhang, Siping Tao, Hebin Yu, and Shu Shi. 2023. TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3613443

*This work was completed before the author joined Meituan.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613443



(a) Previous methods (b) Our method

Figure 1: Previous methods are all based on single-path transmission, through which the video frames may be blocked in poor network conditions (as denoted in red color); our method addresses their shortcomings through multi-path transmission.

1 INTRODUCTION

The rapid advancement of network infrastructure and cloud technology has given rise to numerous ultra-low latency video streaming applications, including cloud gaming, cloud virtual reality(VR), and virtual desktops [25]. These applications utilize remote servers to stream interactive video content to users' portable devices, obviating the need for hardware upgrades. For such applications, an optimal user experience typically requires high-quality and smooth video playback as well as a stable interaction latency (also referred to as motion-to-photon or MTP latency in VR applications) of only 100 ms or even less [28], which presents significant challenges for delivering ultra-low latency video over the best-effort Internet.

Existing methods proposed for optimizing videoconferencing and real-time video streaming, including adaptive bitrate streaming [17, 31], dynamic frame control [7, 19], loss recovery [22, 25] and loss prevention techniques [3, 14], are not as effective for ultra-low latency streaming. This is because most of those solutions aim to mitigate the impact of network interruptions rather than prevent them from occurring.[1] For ultra-low latency video applications, the stringent latency constraints leave too limited buffering capacity to counteract even mild network fluctuations and thus result in frequent video stalls. Our analysis of a large-scale Quality-of-Service(QoS) dataset from a commercially operating cloud gaming platform (§2.1) reveals that over half of users experience video

[1]Although forward error correction(FEC) has been widely adopted in real-time streaming, it can only effectively prevent random packet loss rather than the burst loss in most real-world scenarios [26].

freeze for more than 5.2% of their session time - a far from satisfactory experience.

Nevertheless, our data analysis (§2.2) suggests a potential solution. By simultaneously measuring Wi-Fi and cellular network conditions, we observe that although neither consistently offers stable service, it is rare for both to perform poorly at the same time. Streaming on both Wi-Fi and cellular network paths concurrently allows each path to serve as a backup when the other experiences fluctuations. A multi-path transport framework can substantially reduce video stalls, as long as not all paths are simultaneously congested.

Although multi-path transport has been extensively researched for over a decade [10], an effective multi-path solution for ultra-low latency video streaming remains elusive. Recent studies, such as MP-DASH [11] and XLINK [33] propose novel scheduling algorithms on top of transport layer multi-path protocols like MPTCP [13] and MPQUIC [4] to improve video streaming performance. However, these reliable transport protocols are not designed to discard data from a congested path or adjust sending speed to minimize video stalls. We argue that an ideal multi-path design for ultra-low latency video should integrate video encoding, data allocation, application layer FEC(Forward Error Correction), and network transport all together to address various challenges (§2.3).

In this paper, we present TwinStar, a novel multi-path transport framework for ultra-low latency video streaming. TwinStar's core concept, illustrated in Figure 1, involves dividing the original high-frame-rate video into two independently encoded low-frame-rate substreams and transmitting each substream via separate network paths. If one network experiences fluctuations, the system can seamlessly display the other low-frame-rate substream from the stable network. The key design contributions of TwinStar are as follows:

- Flexible Multi-path Framework (§3.1): TwinStar achieves ultra-low latency multi-path transmission with a configurable framework, that has different modes to cater to diverse performance and cost requirements.
- Joint Rate Control (§3.2): We use separate video encoders to eliminate stream dependency, and propose a joint rate control algorithm to calculate encoding bitrate for both substreams, ensuring consistent video quality for display.
- Bitrate Allocation (§3.3): We implement network-aware bitrate allocation between the two paths to address network dynamics and redistribute excessive bitrates from overloaded paths to underutilized paths.
- Inter-Path FEC Recovery (§3.4): We allocate FEC parities packets to paths separate from media packets, making TwinStar more resilient against network loss than conventional intra-path FEC methods.

We have implemented TwinStar and evaluated its performance on real-world networks. The results indicate that TwinStar significantly outperforms the start-of-art methods with 91% stalling reduction, and 11% quality improvement.

2 MOTIVATION

In this section, we analyze the data from a commercial cloud gaming system to evaluate the real-world performance of single-path streaming and the potential improvements achievable through a multi-path framework. We then discuss the key challenges associated with designing a practical multi-path transport framework.

2.1 Cloud Gaming Performance Analysis

We collaborated with ByteDance's Volcengine[2], and obtained a cloud gaming dataset which was collected in June 2022, spanning one month[3]. The dataset consists of each session's *access type* (Wi-Fi or cellular), *average RTT*, *average Loss Rate*, *average interaction latency* (calculated by measuring the delay between user clicks and corresponding frame displays), *stall ratio* (time spent on the stall events[4]) and other related session information.

We first measure the average RTT and loss rate for each session and presented the results in the form of a cumulative distribution function (CDF) in Figure 2. Two observations can be made: Wi-Fi performs better than cellular networks (as indicated by the curves being further to the left), and a significant number of sessions suffer from poor networking services. Despite the rapid growth of overall network conditions [1, 30], providing stable and satisfactory network services remains a significant challenge. This quality degradation adversely affects high-level user engagement.

To illustrate this further, we calculate the distribution of interaction latency, with the results shown in Figure 3 (top). We found that about 63% of sessions in our dataset experience over 100 ms of interaction latency. We also measure the stall ratio for one week and present the results in Figure 3 (bottom). More than 50% of sessions experience over 5.2% stall ratio. These results are unsatisfactory, given that ultra-low latency users are extremely sensitive to latency and stall events, demonstrating that the existing solution that relies on single-path transmission is insufficient to meet ultra-low latency service requirements.

2.2 Potential Multi-path Gain

We design a new experiment to evaluate the potential improvements a multi-path framework could offer over conventional single-path streaming. We add an active probing module to Volcengine's real-time voice chat service to collect RTT and loss rate information from all available paths every second[5]. This specific application is chosen for three reasons: 1) it already implements multi-path transport and sets up both Wi-Fi and cellular connections at the start; 2) it shares the same edge cloud infrastructure with the previously mentioned

[2]Volcengine is a cloud service platform from ByteDance, offering services including content delivery network, live streaming, real-time communication, cloud gaming, and more.

[3]It is worth noting that the dataset contains no user privacy information. Users are notified before streaming that the service is experimental and the network data is collected for research purpose to improve user experience. Users need to give explicit approval for streaming to start.

[4]The stall event is recorded when the gap between consecutive frames is larger than a certain threshold th. We set th = 100ms in this paper. Other th settings are also acceptable and will not lead to significant differences.

[5]By default the multi-path transport feature remains off for all users. The probing module is only activated when the user explicitly checks the setting box to enable the multi-path transport feature, which incurs cellular data cost and gives permission to use network data for research purpose.

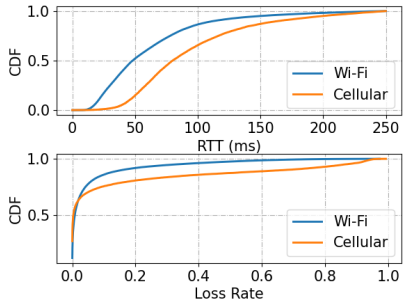


Figure 2: The RTT and loss rate CDF for Wi-Fi and cellular networks.

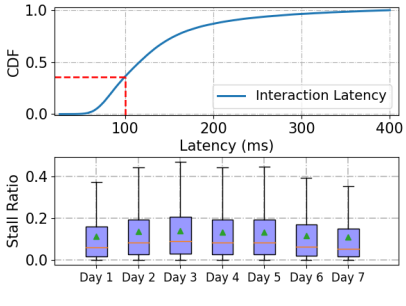


Figure 3: The interaction latency and stall ratio of online cloud-gaming users.

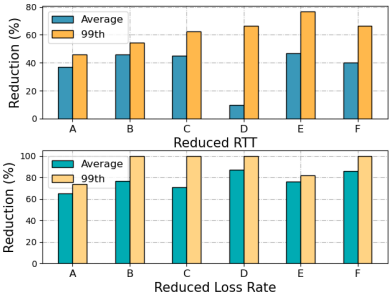


Figure 4: Potential improvement when using multi-path transmission

cloud gaming system; 3) the voice chat app is lightweight and capable of recording network information at a finer granularity.

In total, we collect 1.6 million sessions covering 1.3 million unique users from more than twenty countries. We use this probing dataset for a simple analysis to estimate multi-path potential benefits. We compare two analysis methods: **Single-best** (upper-bound of single-path transmission), which analyzes both Wi-Fi and cellular networks and selects the best path to send all frames, and **Multi-best**, which dynamically selects the best path for each frame. In practice, for each session, we calculate the average value of RTT and loss rate for both Wi-Fi and cellular and select the better value of the two (in this case, the lower value is better) to represent the **Single-best** value of the session. To calculate the **Multi-best** value, we compare at the second level and only picked the better value between Wi-Fi and cellular to calculate the session average.

We choose the top six countries with the most users and label them from A to F. For each country, we first use the session average value to calculate the country's average and 99th percentile and then plotted the reduction percentage of **Multi-best** over **Single-best** in Figure 4. The results indicate that **Multi-best** performs much better than **Single-best** in both RTT and loss rate, with consistent improvements across different countries. Moreover, **Multi-best**'s outperformance is particularly significant for optimizing tail performance (E.g. **Multi-best** achieves 38% improvement in reducing average RTT, but over 44% for 99-th percentile RTT of country *A*). These observations demonstrate that while Wi-Fi and cellular networks may experience fluctuations, it is uncommon for both network paths to perform poorly simultaneously.

2.3 Multi-path Challenges

Although multi-path transmission has the potential to provide better QoS, simply duplicating single-path to multi-path is inefficient. We outline three key design challenges for a practical multi-path streaming framework.

Design Challenge #1: *stream dependency.* Mere use of transport layers multi-path protocols, such as MPTCP or MPQUIC, does not work effectively for ultra-low latency video due to path heterogeneity and frame inter-dependence. This approach can cause Head-of-Line (HoL) blocking, where frames from the faster path must wait for those from the slower path to ensure reliability. To avoid HoL blocking, it is crucial to ensure the video streams of each path can be independently decoded. While copying the same video stream for each path, like Re-MP [8], solves the dependency problem, it results in significant overhead (i.e., 100% redundancy). Our design goal is to generate independent video streams using less bandwidth than Re-MP.

Design Challenge #2: *bitrate allocation.* Conventionally, real-time video streaming dynamically updates the encoding video bitrate to match the available network bandwidth. When network conditions deteriorate, the streaming application typically reduces the video encoding bitrate to prevent network congestion. However, in the multi-path scenario, it is not necessary to cut the video bitrate if the transport framework can dynamically allocate excessive bits to the uncongested path so that the video quality maintains. Furthermore, the multi-path framework may need to support flexible bitrate allocation strategies to maintain the advantages of multi-path as well as meet application need, such as minimizing cellular data usage particularly in areas where the cost of cellular data remains high.

Design Challenge #3: *data recovery.* Although using multiple paths prevents video stalls when only one path losses packets, the streaming framework still needs to improve the efficiency of data recovery because it requires data from all paths to deliver the best quality video. FEC is widely used as a recovery method to overcome packet loss in ultra-low latency applications. However, high FEC packet volume can decrease video encoding bitrate when both types of packets share the same path. Existing data recovery strategies typically respond to high packet loss rates by increasing the number of FEC packets, which may worsen network conditions. One advantage of a multi-path framework is that it can use another path to pass recovery data, such as XLINK [33], which uses a different path for retransmission. The multi-path framework design should also consider adding inter-path recovery to further improve streaming performance.

3 SYSTEM DESIGN

In this section, we present TwinStar. Our design aims to meet the following goals: (1) eliminate HoL blocking between different paths with less than 100% overhead; (2) support dynamically allocating

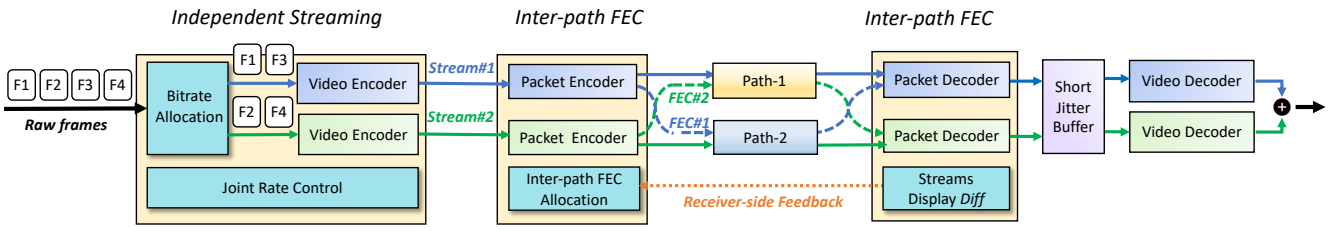


Figure 5: The framework overview and workflow of TwinStar.

excessive data from congested paths to transmit via underutilized paths; (3) enable inter-path data recovery to outperform intra-path recovery efficiency; and (4) make the framework flexible for real-world use.

3.1 Overview

We present the design of TwinStar framework in Figure 5. The server sends the original uncompressed video stream to the bitrate allocation module, which divides the frames with 1:1 ratio into two separate streams and encodes them using different video encoders. Encoded Stream #1 and Stream #2 are then transmitted over Wi-Fi and cellular paths, respectively. Additionally, each stream is protected by an individual FEC encoder that generates and transmits protection packets using different paths. On the receiver side, Stream #1 and Stream #2 are decoded separately and then merged to display the final output. If any path congestion occurs (e.g., if Stream #1 on the Wi-Fi path experiences congestion), the decoded video from the other path (i.e., Stream #2 on the cellular path) can be displayed directly with a reduced frame rate without blocking. The frame rate will soon resume by increasing recovery packets from the non-congested path, based on feedback from the receiver (i.e., FEC#1). Moreover, the short jitter buffer on the receiver side allows the faster stream to wait a brief period for the slower stream to catch up, preventing frame drops.

TwinStar also allows applications to configure the framework to better meet their specific needs. For example, if users are sensitive to cellular data cost, the application provider may want to minimize bandwidth consumption on the cellular path. In such cases, TwinStar can be configured to operate in a bandwidth-saving mode, which only encodes the full video as one stream (using one encoder and decoder) and transmits media data over the Wi-Fi path. By doing so, the bandwidth cost on the cellular path is minimized, as it only needs to transmit inter-path FEC traffic when the Wi-Fi path experiences packet loss and high RTT. This mode can still outperform single-path transmission, as it can assist the Wi-Fi path in data recovery without affecting the original video stream (as demonstrated in §5).

For the rest of the section, we will introduce in details three key modules in TwinStar's design: joint rate control, network-aware bitrate allocation, and inter-path FEC recovery.

3.2 Joint Rate Control

To eliminate the stream dependency between two paths, we use two individual video encoders to encode the video frame allocated to each path. These substreams can be transmitted and displayed independently without head-of-line blocking or merged together to aggregate the bandwidth of multipath.

However, merely utilizing two default real-time encoders to generate independent streams may lead to quality fluctuations, which can adversely impact the user experience when merging two streams. Several factors can lead to such quality differences. Firstly, real-time video encoders often employ the variable bitrate rate control (VBR) algorithm that adjusts the encoding bitrate based on the path-level estimated bandwidth. In an environment with heterogeneous bandwidth, the encoding bitrate of the two encoders may differ significantly. Secondly, the complexity of the source video put into each encoder may vary in terms of frame content and frame rate. Even if both encoders use the same bitrate, the encoded picture quality between them may differ.

To solve the problem, we use a framerate-bitrate model for joint rate control, which enables two encoders to synchronize encoding quality. The model is first employed in [16] and illustrated as Eq.(1), where t is the input framerate, t_{max} is the max framerate of the video source, and the normalized bitrate $R_t(t) = R(t) \,/\, R(t_{max})$. The parameter b indicates how fast the bitrate decreases with a reduction in framerate, while maintaining consistent video quality. With this model, we can estimate the bitrate relationship between the two encoders and calculate the target bitrate of each stream that satisfies their bandwidth restriction.

$$R_t(t) = (t/t_{max})^b \tag{1}$$

In the offline stage, we use a large number of video sequences covering different scenarios to train the model parameter b. We have found that the model fits very well and remains stable in most situations. This means that we can use well-trained parameters in the online system for quality synchronization. At runtime, the model is integrated to update the two encoders by taking the frame rate allocation as input and the target bitrate for two streams as output.

3.3 Network-aware Bitrate Allocation

The bitrate allocation module is responsible for dividing video traffic into two paths. By decoupling data allocation and encoding, we can enable flexible video split strategies. By default, TwinStar

uses a simple 1:1 round-robin split strategy, which allocates frames evenly to two video encoders[6]. This strategy is easy to implement and can provide at least half of the full frame rate to avoid stalls, regardless of which path becomes congested. Our experimental results demonstrate that, despite using such a simple strategy, we still outperform state-of-the-art methods (as shown in §5).

The bitrate allocation also needs to consider network dynamics. When the estimated bandwidth of a path deteriorates and the encoding rate drops, the encoding rate of the other high-bandwidth path is also decreased in order to match the synchronized quality. This could result in low bandwidth utilization and unnecessary low encoding quality.

The fundamental idea of TwinStar is to dynamically offload excessive bits to the uncongested path so that the video quality maintains. First, we use a bandwidth threshold θ to identify the low-bandwidth scenario. When a path gets congested and the estimated bandwidth decrease to lower than θ, we check whether there exists idle bandwidth resource on another path. If there is, then the constrained path will 1) use adjusted bandwidth constraint (i.e. local-path bandwidth plus left bandwidth from another path) for joint rate control; 2) randomly drop part of frames from encoded sub-stream to ensure transmission data do not exceed local-path capacity; 3) increase the ratio of inter-path FEC on the dropped frame to 100% to ensure the recovery of dropped frames.

3.4 Inter-path FEC Recovery

Inter-path FEC Allocation. TwinStar follows a basic recovery principle of transmitting FEC packets through a different path from the media data to prevent simultaneous stream failure. For instance, if the Wi-Fi path becomes congested, neither FEC packets nor media packets can be received in intra-path FEC protection methods. This ultimately leads to the failure of stream #1. In contrast, with inter-path FEC protection methods, even if media packets are congested in the Wi-Fi path, stream #1 can be recovered by using the FEC packets from the cellular path.

Specifically, TwinStar needs to decide how many FEC packets should be generated and how to allocate them within the available bandwidth. The FEC allocation strategy is illustrated in Algorithm 1. Firstly, for the calculation of the FEC ratio, we follow the approach used in WebRTC, which is based on the packet loss rate. The difference is that we use the packet loss rate and media packet count of the other path to calculate the FEC count of the current path (lines 1-4). Note that if low-bandwidth offloading is triggered, the relative inter-path FEC ratio will be directly set to 100%. Secondly, when transmitting FEC packets across paths, we also need to consider the bandwidth allocation between FEC and media data. If both paths have sufficient bandwidth, the FEC data is transmitted across paths directly (as in lines 5-6). If one path has insufficient bandwidth but the other is sufficient, the FEC payload is offloaded into the sufficient path (as in lines 7-14). If both paths have insufficient bandwidth, we adjust the bitrates of the two streams to ensure that the total data volume does not exceed the available bandwidth (as in lines 15-17).

[6]In future work, we will explore the integration of more complex split strategies (such as the classic scheduling algorithm in [13]) into TwinStar.

Algorithm 1: Inter-path FEC Allocation Algorithm

Input: $bw_1, bw_2, loss_1, loss_2, mediabr_1, mediabr_2$
Output: $fecbr_1, fecbr_2$

1 $fecbr_1 = mediabr_1 \cdot loss_1$
2 $fecbr_2 = mediabr_2 \cdot loss_2$
3 $\Delta_1 = bw_1 - (mediabr_1 + fecbr_2)$
4 $\Delta_2 = bw_2 - (mediabr_2 + fecbr_1)$
5 **if** $\Delta_1 > 0$ *and* $\Delta_2 > 0$ **then**
6 | return;
7 **else if** $\Delta_1 > 0$ *and* $\Delta_2 \leq 0$ *and* $|\Delta_2| < \Delta_1$ **then**
8 | $fecbr_1 = bw_2 - mediabr_2$
9 | $fecbr_2 += |\Delta_2|$
10 **end**
11 **else if** $\Delta_2 > 0$ *and* $\Delta_1 \leq 0$ *and* $|\Delta_1| < \Delta_2$ **then**
12 | $fecbr_2 = bw_1 - mediabr_1$
13 | $fecbr_1 += |\Delta_1|$
14 **end**
15 **else**
16 | inform rate control to adjust $mediabr_1, mediabr_2$
17 **end**

Receiver-feedback FEC Adjustment. Relying solely on the sender's state to determine the FEC ratio may not achieve the expected effect in real-world scenarios. Since network bursts and dynamics are often unpredictable, once the number of FEC packets falls below the number of lost packets, all FEC protection will fail [21, 25]. Fortunately, we have observed that under the multi-path framework, the states of the two streams can provide good guidance for FEC adjustment. For example, if stream #1 significantly falls behind stream #2, it means that it is congested and its FEC protection is not sufficient for recovery. The sender should increase the FEC ratio of stream #1 until the playback progress of the two streams is equal. This insight allows us to use the *difference in playback progress between the two streams* as a compass for FEC recovery efficiency perception and adjustment.

Overall, we design a double-threshold dynamic adjustment mechanism, which works as follows:

- The receiver periodically updates the playback progress difference between the two streams and sends a notification to the sender. The metric is calculated as the number of video frames that lag behind in the late stream.

- The sender uses two thresholds to adjust the FEC ratio based on receiver-side feedback. i) If the difference in playback progress is greater than threshold α, the sender increases the FEC ratio of the lagging stream to provide full protection; ii) If the difference decreases to lower than threshold β, the sender decreases the FEC ratio of the lagging stream to the baseline value, which is calculated by the sender based on the loss rate of the paths. Parameters α and β can be assigned using empirical values.

4 IMPLEMENTATION

Multipath Extensions. TwinStar extends WebRTC to support multipath transport for real-time video based on [29]. In particular, TwinStar uses a separate space to allocate packet sequence numbers in different paths. This enables the receiver to determine subpath-related packet jitter and packet loss. Similar to single transmission, sub-path transmission in TwinStar uses negative acknowledgment (NACK) for loss detection and packet retransmission. Additionally, we introduce a two-level Real-Time Control Protocol (RTCP). The top-level RTCP reports receiver-side feedback to support FEC adjustment. The second-level RTCP reports separately by two paths, which enables individual sender reports and receiver reports to maintain path-level congestion control.

Video Encoders and Decoders. To enable correct frame compression and displaying, TwinStar use a separate frame sequence space (i.e. local frame ID) by each encoder for video decoding, and use a global frame sequence space (i.e. global frame ID) for video display. The local frame ID can help decoders find correct reference relationships and decode frames. The global frame ID is useful in finding the correct displaying order and discarding outdated frames.

FEC Encoders and Decoders. To enable inter-path FEC, TwinStar move the FEC packet encoding and decoding module from the subpath transport layer to the application layer. When video frames are compressed, the FEC encoding module will generate frame-level protection packets. By using the global frame ID as a grouping key, FEC packets can be delivered with any paths and can be successive to recover. The FEC decoder in the receiver collects FEC packets from all network pipelines, mapping them with global frame ID, and transfers recovered media packets to relative video decoders.

Short Jitter Buffer. The buffer performs two key functions: first, NACK can wait for a short period of time for the arrival of the FEC packet from the other path instead of an immediate request. Second, a received frame can wait for a short time instead of directly sending to the decoder, to avoid the next frame from the other path becoming outdated and discarded. For ultra-low latency, we set the buffer length to the 100ms threshold minus current interaction latency, as exceeding 100ms might affects user experience.

Path Management. Besides connectivity, TwinStar put a requirement of latency on path availability. Specifically, TwinStar requires the RTT difference of two paths no greater than the length of the short jitter buffer. If the delay requirement is not met, TwinStar will revert to a single-path transmission. Since the slower stream will always be late and discarded by the receiver, streaming such a stream will not only waste bandwidth but also incurs unnecessary codec resources.

Integration with Applications and CDN Server A TwinStar client is written in C++ with 13k LoC, including video decoders, FEC decoders and short jitter buffer, which is integrated into an Android App. TwinStar offers APIs for applications to pass parameters(e.g. mode). A TwinStar server is also written in C++ with 20k LoC including video encoders and FEC encoders and related algorithms, which is deployed in commercial servers equipped with hardware codecs.

5 EVALUATION

5.1 Methodology

Baselines. We compare TwinStar with the following three baselines, all of which are implemented on top of WebRTC:

▹ Single-path (denoted as *Single*) [9]: This algorithm represents the state-of-art single-path streaming, which utilizes multiple recovery techniques to handle packet loss, such as NACK, FEC and IDR. By default, Wi-Fi access takes priority over cellular network access for users.

▹ Vanilla Multi-path (denoted as *Vanilla-MP*): This algorithm represents the state-of-art multi-path streaming with a default MinRTT scheduler and traditional intra-path recovery techniques. The implementation of *Vanilla-MP* is mainly based on the design principles of MPTCP [13, 24].

▹ ReMP [8]: This algorithm duplicates each packet, and sends them through two paths. Although this method requires a 100% redundancy and is inappropriate to serve online users, it can be used to evaluate the improvement space of multi-path transmission on loss rate and RTT reduction.

Performance Metrics. The comparison consists of two parts, which are QoE and bandwidth cost. For QoE, we consider two industrial standard metrics including *stall ratio* (denoted as ***Stall***) and *Peak Signal-to-Noise Ratio* (denoted as ***PSNR***). *Stall* cares about the time spent on stall event; *PSNR* represents the video quality. For the cost, we compare the bandwidth costs when all baselines achieve the same PSNR.

Environmental Setup. Through cooperation with the cloud-gaming platform, we have deployed TwinStar in the real world and conducted a large-scale test spanning one week (from March 1st to 8th, 2023). The users come from five cities (denoted as D_1 to D_5) in China, and all of them are equally divided to run with different algorithm.

5.2 Results and Discussion

QoE Gains: We first analyze QoE, and Table 1 shows the average experimental results of all sessions (the values in parentheses represent the 95th percentile stall ratio and 5th percentile PSNR, respectively). In terms of stall ratio, TwinStar significantly outperforms *Single* and achieves results comparable to the improvement space of multipath transmission (i.e. *ReMP*). Take D_1 as an example, TwinStar reduces average stall by 91% and reduces 95th percentile stall by 93%, leveraging multipath transmission and HoL blocking elimination to effectively reduce stall events. We confirm that the outperformance of TwinStar is consistent among all regions, and becomes more significant for tail optimization. However, despite utilizing multiple paths, *Vanilla-MP* fails to decrease stall and performs even worse than *Single*. For example, in D_2, there is a 7.4% increase in average stall ratio and a 24% increase in 95th percentile stall ratio. Due to its susceptibility to HoL blocking and poor performance in highly dynamic networks [33], *Vanilla-MP* results in a higher tail stall than *Single* in ultra-low latency streaming.

In terms of PSNR, TwinStar achieves the best performance among all baselines. For example, in the region D_1, TwinStar improves the

Table 1: The evaluation on real-world deployment.

	Single		Vanilla-MP		TwinStar-S		TwinStar		ReMP	
	Stall (%)	PSNR (dB)	Stall (%)	PSNR (dB)	Stall (%)	PSNR (dB)	Stall (%)	PSNR (dB)	Stall (%)	PSNR (dB)
D_1	8.3 (21.3)	28.4 (24.8)	9.3 (27.5)	28.2 (25.8)	3.1 (8.9)	28.8 (26.4)	0.7 (1.5)	31.7 (28.6)	0.4 (1.1)	28.9 (26.5)
D_2	8.1 (20.4)	28.3 (24.6)	8.7 (25.3)	28.5 (26.7)	3.3 (9.2)	28.5 (26.2)	0.8 (1.6)	31.8 (28.5)	0.5 (1.4)	28.4 (26.1)
D_3	6.1 (15.9)	28.2 (24.1)	7.1 (17.3)	28.7 (26.8)	3.0 (8.9)	29.1 (26.7)	0.6 (1.1)	31.5 (28.1)	0.2 (0.4)	29.1 (26.8)
D_4	7.5 (19.8)	28.7 (24.6)	9.0 (26.7)	28.2 (26.1)	3.1 (9.0)	28.9 (26.5)	0.8 (1.8)	31.4 (28.2)	0.3 (1.0)	28.7 (26.4)
D_5	7.2 (16.4)	28.2 (24.2)	8.6 (20.7)	27.6 (25.3)	2.7 (8.5)	29.3 (26.8)	0.4 (0.7)	31.6 (28.4)	0.2 (0.6)	28.9 (26.6)

average and 95th percentile PSNR by 12% and by 15% respectively, as compared to *Single*. What's more, TwinStar also significantly outperforms the second-best baseline (i.e., *ReMP*). This is rational: although both TwinStar and *ReMP* leverage multi-path transmission without path dependency, however, through video splitting and sub-stream encoding instead of copying, TwinStar can aggregate the bandwidth of two paths and therefore higher bitrate.

At the same time, we also demonstrate the configurability of TwinStar. Specifically, it is set to the bandwidth-saving mode that only uses the Wi-Fi path to send video data and uses cellular path to send inter-path FEC. The results are also presented in Table 1 (denoted as *TwinStar-S*). We can see that *TwinStar-S* still significantly outperforms Single in all QoE metrics. Across all regions, *TwinStar-S* achieves a 63% reduction on average stall and 2% improvement on average PSNR as compared to *Single*. Meanwhile, as shown in Table 2, *TwinStar-S* effectively reduces data usage[7] on the cellular path to as low as 17% of the Wi-Fi path.

Table 2: Normalized data usage of TwinStar and TwinStar-S.

	Wi-Fi	Cellular	Total
TwinStar	0.568	0.432	1.000
TwinStar-S	0.547	0.096	0.643

Bandwidth Cost: We assess bandwidth costs by comparing the normalized bandwidth cost of different algorithms, using a target PSNR. In dynamic networks, maintaining consistent encoding quality across systems is difficult. Therefore, we evaluate the amount of video data that needs to be transmitted under a fixed PSNR setting, without considering the cost of FEC data. We use four different games, i.e. Stickfight, Beheweled, Honor of King, and Maplestory. The results are shown in Figure 6. From the results, we can derive the following findings. First of all, *Single* and *Vanilla-MP* achieve the lowest bandwidth costs by employing non-redundant data transmission, resulting in a bandwidth cost equal to the media bitrate. Second, the bandwidth cost of TwinStar is slightly higher than *Single* (from 1.08× to 1.32×), but significantly lower than *ReMP*. This is rational: on the one hand, TwinStar uses two independent encoders, which will decrease the encoding efficiency (more I frames and increasing the difference between frames); on the other hand, as *ReMP* duplicates packets on all paths, it will need 2× traffic to the media bitrate. Considering the benefits we obtain in terms of QoE, we believe that the bandwidth overhead is acceptable.

[7]Data usage is normalized by dividing the maximum element in the table (i.e., the total data usage in TwinStar).

5.3 Framework Deep Dive

We conduct the ablation study and illustrate the necessity of each component in TwinStar's design. The experiment is performed in a controlled server where we can configure the path characteristics.

HoL Blocking: We first demonstrate that our two-encoder design effectively eliminates the HoL blocking problem. We collected cellular and Wi-Fi traces in a shopping mall and conducted a trace-driven evaluation using Mahimahi emulation tools [20]. One of the traces is plotted on the right-hand side of Figure 7, from which we can observe that the latency of both paths is highly variable. We compare TwinStar and TwinStar with only one encoder, and focus on the length of blocked frames in the receiver buffer. The results are presented on the left-hand side of Figure 7. We observed that TwinStar with a single encoder cannot handle path heterogeneity well and its queue length keeps growing as δ_{RTT} increases. In contrast, TwinStar consistently performs well even with high δ_{RTT}, which can also explain the low stall ratio observed in §5.2.

Rate Control: Furthermore, we demonstrate the necessity of the joint rate control when merging streams from two paths. We measure the variation in PSNR (denoted as δ_{PSNR}) between two adjacent frames to indicate quality fluctuation, where a smaller δ_{PSNR} indicates better stability. We conduct experiments on four games, and Figure 8 shows the δ_{PSNR} of TwinStar with and without joint rate control. We observed that without the rate control algorithm (i.e., when the bitrate of each stream is set to the available bandwidth of that path), the merged stream suffers from significant PSNR variation of 3-6 dB on average, which can negatively impact the user's viewing experience. In contrast, the joint rate control algorithm maintains an average δ_{PSNR} of less than 0.2 dB, resulting in a much more stable stream and higher user engagement.

Recovery Efficiency: Finally, we evaluate the impact of inter-path FEC design on video bitrate and rendering frame rate. Bitrate evaluates the rationality of FEC allocation, while frame rate reflects on recovery effectiveness. We use frame rate instead of stall ratio here because TwinStar uses two encoders, and therefore, the failure of FEC recovery for one stream may not necessarily result in a stall event. We set a 4Mbps bandwidth limit for both Wi-Fi and cellular paths and introduce varying loss rates on the Wi-Fi path (the top sub-figure in Figure 9). We observed that the bitrate and frame rate with inter-path FEC are both higher than with intra-path FEC (the middle and bottom sub-figures). This can be explained by our inter-path design, which can effectively offload FEC traffic to the more abundant cellular path when Wi-Fi has encountered high loss rates (i.e., after 70s). The feedback-based FEC adjustment enables

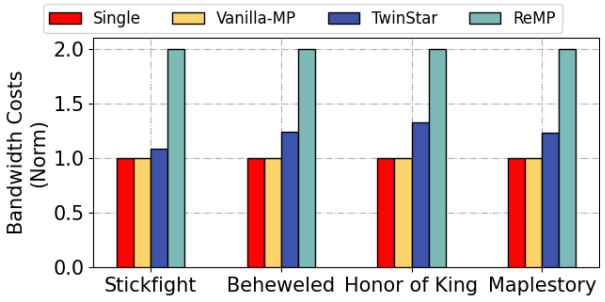


Figure 6: The comparison of bandwidth cost.

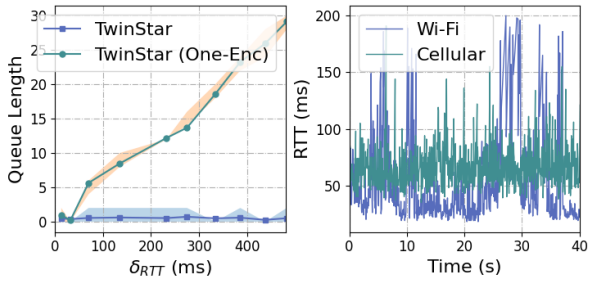


Figure 7: The necessity of two encoders.

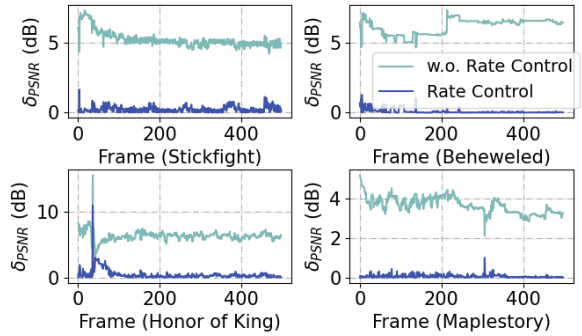


Figure 8: The necessity of rate control algorithm.

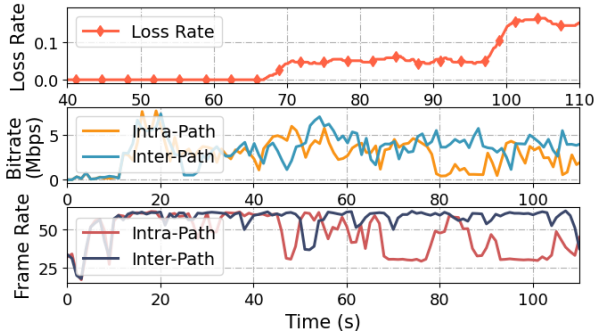


Figure 9: The necessity of inter-path recovery.

TwinStar to quickly resume the damaged stream (i.e., Wi-Fi stream) and maintain a high frame rate for most of the time.

Computation Overhead: We also evaluated the computational overhead of using two encoders and decoders. Table 3 presents the CPU utilization and power consumption of TwinStar-S, which employs one encoder/decoder instance, and TwinStar, which uses two. The power consumption data of the server has been omitted because the server is constantly powered on. The results show that the CPU utilization for TwinStar on the server increased by only 3.30%, and the power consumption on the client increased by 2.72% This is because, even though two encoder and decoder instances were used, the total number of video frames remained the same.

Table 3: CPU utilization and power consumption.

	Client		Server
	CPU Uti. (%)	Power (mW)	CPU Uti. (%)
TwinStar	16.07	3092.09	10.24
TwinStar-S	15.54	3008.05	9.84

6 RELATED WORK

Ultra-low Latency Video Streaming Optimization. Many optimizations have been proposed to enhance the QoE of ultra-low latency video streaming. Some focus on FEC[2, 15, 23], which transmit redundant packets to achieve fast data recovery from loss events. Some works, such as scalable video coding(SVC), have designed special codec modes that support layer coding[12, 27]. Some works try to recover from data loss by leveraging the independent data refresh (IDR) frames[25] which can be decoded independently. Other works, also focus on bitrate adaption[7, 32], frame rate control[19] and middlebox modification[18] to improve the overall performance. Although all of the above methods improve transmission efficiency to some extent, they are limited by single-network resources, especially when the network is in bad condition.

Multi-networking Transmission for Video Streaming. MPTCP was standardized by IETF in 2013[6, 24]. By enabling end hosts to leverage multiple NICs, MPTCP greatly improves transmission efficiency. Based on MPTCP, MP-DASH[11] introduces support for the DASH framework and it can provide better support for video services by taking interface preferences from users. Different from MP-DASH's tight integration with TCP, the userspace property of MPQUIC provides a more flexible design space[4, 5]. However, existing schemes still suffer from the hair-trigger HoL blocking, leading to a poor performance in ultra-low latency video streaming.

7 CONCLUSION

We have described TwinStar, a flexible multipath framework for ultra-low latency video delivery. Observing that the probability of two paths simultaneously encountering issues is much lower than that of a single path, TwinStar split video into two independent substreams on distinct paths to avoid stalling. We show that with minor bandwidth cost compared to single-path transmission, TwinStar improve the QoE by 91% reduction on stall ratio and 11% improvement on PSNR.

REFERENCES

[1] Serhat Arslan, Yuliang Li, Gautam Kumar, and Nandita Dukkipati. 2023. Bolt: Sub-RTT Congestion Control for Ultra-Low Latency. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 219–236. https://www.usenix.org/conference/nsdi23/presentation/arslan

[2] Ke Chen, Han Wang, Shuwen Fang, Xiaotian Li, Minghao Ye, and H Jonathan Chao. 2022. RL-AFEC: adaptive forward error correction for real-time video communication based on reinforcement learning. In *Proceedings of the 13th ACM Multimedia Systems Conference*. 96–108.

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

[4] QD Coninck and Olivier Bonaventure. 2020. Multipath Extensions for QUIC (MP-QUIC). *IETF, Individual Submission, Internet Draft draftdeconinck-quic-multipath-04* (2020).

[5] 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*. 160–166.

[6] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. 2013. *TCP extensions for multipath operation with multiple addresses*. Technical Report.

[7] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 267–282.

[8] 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.

[9] Google. 2018. WebRTC Home. https://webrtc.org/.

[10] Sana Habib, Junaid Qadir, Anwaar Ali, Durdana Habib, Ming Li, and Arjuna Sathiaseelan. 2016. The Past, Present, and Future of Transport-Layer Multipath. *Journal of Network and Computer Applications* 75 (01 2016). https://doi.org/10.1016/j.jnca.2016.09.005

[11] 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*. 129–143.

[12] Yi-Mao Hsiao, Chia-Hsiang Chen, Jeng-Farn Lee, and Yuan-sun Chu. 2012. Designing and implementing a scalable video-streaming system using an adaptive control scheme. *IEEE Transactions on Consumer Electronics* 58, 4 (2012), 1314–1322.

[13] IP Networking Lab. 2019. Linux Kernel Implementation. https://multipath-tcp.org/.

[14] Insoo Lee, Seyeon Kim, Sandesh Sathyanarayana, Kyungmin Bin, Song Chong, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2022. R-FEC: RL-based FEC Adjustment for Better QoE in WebRTC. In *Proceedings of the 30th ACM International Conference on Multimedia*. 2948–2956.

[15] Yanlin Liu and Mark Claypool. 1999. Using redundancy to repair video damaged by network data loss. In *Multimedia Computing and Networking 2000*, Vol. 3969. SPIE, 73–84.

[16] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. 2012. Modeling of Rate and Perceptual Quality of Compressed Video as Functions of Frame Rate and Quantization Stepsize and Its Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 22 (05 2012), 671–682. https://doi.org/10.1109/TCSVT.2011.2177143

[17] 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*. 197–210.

[18] 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*. 193–206.

[19] Zili Meng, Tingfeng Wang, Yixin Shen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2023. Enabling High Quality Real-Time Communications with Adaptive Frame-Rate. In *USENIX NSDI*.

[20] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference* (Santa Clara, CA) *(USENIX ATC '15)*. USENIX Association, USA, 417–429.

[21] Colin Perkins and Orion Hodson. 1998. *Options for repair of streaming media*. Technical Report.

[22] Dimitri Podborski, Yago Sanchez, Robert Skupin, Cornelius Helige, and Thomas Schierl. 2017. Tile based panoramic streaming using shifted IDR representations. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 565–570.

[23] Rohit Puri, Kannan Ramchandran, Kang-Won Lee, and Vaduvur Bharghavan. 2001. Forward error correction (FEC) codes based multiple description coding for Internet video streaming and multicast. *Signal Processing: Image Communication* 16, 8 (2001), 745–762.

[24] 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)*. 399–412.

[25] Devdeep Ray, Vicente Bobadilla Riquelme, and Srinivasan Seshan. 2022. Prism: Handling Packet Loss for Ultra-low Latency Video. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3104–3114.

[26] Michael Rudow, Francis Y. Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and K.V. Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 953–971. https://www.usenix.org/conference/nsdi23/presentation/rudow

[27] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.

[28] Shu Shi, Varun Gupta, and Rittwik Jana. 2019. Freedom: Fast recovery enhanced vr delivery over mobile networks. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 130–141.

[29] Varun Singh, Saba Ahsan, and Jörg Ott. 2013. MPRTP: multipath considerations for real-time media. In *Proceedings of the 4th ACM Multimedia Systems Conference*. 190–201.

[30] Ping Yang, Yue Xiao, Ming Xiao, and Shaoqian Li. 2019. 6G wireless communications: Vision and potential techniques. *IEEE network* 33, 4 (2019), 70–75.

[31] 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*. 325–338.

[32] 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*. 775–788.

[33] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. 2021. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*.