XLINK: QoE-Driven Multi-Path QUIC Transport in Large-scale Video Services

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[†], Ming Zhang[†]

[†]Alibaba Group [§]ICT-CAS

[*]Co-first authors

ABSTRACT

We report XLINK, a multi-path QUIC video transport solution with experiments in Taobao short videos. XLINK is designed to meet two operational challenges at the same time: (1) Optimized user-perceived quality of experience (QoE) in terms of robustness, smoothness, responsiveness, and mobility and (2) Minimized cost overhead for service providers (typically CDNs). The core of XLINK is to take the opportunity of QUIC as a user-space protocol and directly capture user-perceived video QoE intent to control multi-path scheduling and management. We overcome major hurdles such as multi-path head-of-line blocking, network heterogeneity, and rapid link variations and balance cost and performance.

To the best of our knowledge, XLINK is the first large-scale experimental study of multi-path QUIC video services in production environments. We present the results of over 3 million e-commerce product short-video plays from consumers who upgraded to Taobao android apps with XLINK. Our study shows that compared to single-path QUIC, XLINK achieved 19 to 50% improvement in the 99-th percentile video-chunk request completion time, 32% improvement in the 99-th percentile first-video-frame latency, 23 to 67% improvement in the re-buffering rate at the expense of 2.1% redundant traffic.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Cross-layer protocols**.

KEYWORDS

QUIC, Multi-path, Wireless Transport, QoE, Scheduling, Video

ACM Reference Format:

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[†], Ming Zhang[†]. 2021. XLINK: QoE-Driven Multi-Path QUIC Transport in Large-scale Video Services. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–28, 2021, Virtual Event, USA*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3452296.3472893

SIGCOMM '21, August 23–28, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8383-7/21/08…$15.00
https://doi.org/10.1145/3452296.3472893

1 INTRODUCTION

Video streaming (either live or video-on-demand) has now become the central component in today's e-commerce. Ninety-six percent of the consumers report that product videos are incredibly helpful in their purchasing decisions [1]. The current COVID-19 pandemic is further accelerating such transitions of consumers' habits in viewing and buying, pushing the need for more sellers to create video content on platforms such as Alibaba, Amazon, YouTube, and TikTok and spawning a wave of Internet celebrity economy [2] [1].

As one of the world's primary e-commerce services, we make two key observations: first, short-form product video stalls and start-up delays significantly impact user satisfaction. Second, the need driven by consumers who want to see greater detail and feel more engaged keeps pushing videos towards higher data rates. These observations urge the demand for more wireless bandwidth and more robust data delivery to mobile devices that may experience frequent hand-offs. Multi-path transport that enables multi-homed devices, like smartphones, to aggregate wireless links is a promising solution to meet the above need.

Indeed, the topic of multi-path transport has gained much attention over the past few decades [4–8]. Today, the most widely known multi-path protocol is MPTCP in RFC 6824 [4]. Unfortunately, MPTCP requires OS-level support in smartphones. The deployment cost is exceptionally high for mobile app providers who are not phone manufacturers [2]. Recently, as the industry moves toward QUIC [10], multi-path extensions over QUIC [7, 11, 12] have been introduced. In contrast to TCP, QUIC as a user-space protocol, can be installed and upgraded as part of the application, so multi-path QUIC as an end-to-end solution can be deployed rapidly and evolved continuously [10]. However, recent proposals [7, 13] were designed to support generic traffic such as web and bulk data transfer and were not optimized for videos. The effectiveness of these proposals in practice also remains unclear [14, 15].

In this paper, we ask the following question: *is it practical and worthwhile to bring multi-path QUIC to large-scale video services?* We found it was far from straightforward to apply past multi-path solutions into our large-scale product video services directly. First

[1] For example, Ryan Kaji, who was nine years old as of 2020, had influenced the toy industry with his YouTube channel, Ryan's World. Ryan was also listed as the highest paid YouTuber, earning $22 million and $26 million respectively from his videos and product line [3]

[2] There are over 1000 smartphone models that need to be covered [9].

of all, to justify the incentives, multi-path should achieve no worse performance than a single path [6, 16]. However, our deployment showed that the default multi-path solutions could be 16% slower on the median and 28% slower at 99-percentile than the single-path delivery over the better path. The efficient usage of the aggregated wireless resources turns out to be much harder than expected. One of the major hurdles is the multi-path head-of-line (MP-HoL) blocking issue caused by fast varying and heterogeneous paths [6]. The blocking happens when the packets sent earlier at the slow path arrive later than the packets at the fast path, causing out-of-order arrival; the out-of-order packets are not eligible to be submitted to applications, so the fast paths have to wait. Significant heterogeneity over Wi-Fi, LTE, and 5G, as well as frequent handoffs of mobile terminals in our context, will further aggravate this issue [17].

One solution to the multi-path HoL blocking is to use more sophisticated packet scheduling algorithms [18–20]. These solutions, however, rely on predictions of network characteristics, which tend to be inaccurate, especially for wireless links that experience rapid varying link rates and occasional multi-second outages [21]. Other proposals attempt to transmit duplicate packets to decouple multi-paths to alleviate the problem [6, 8]. Unfortunately, they cause significant traffic redundancy. For instance, we found that the traditional re-injection strategies [6] lead to a more than 15% increase of the total outbound traffic from video servers in practice. The cost overhead is particularly important when it comes to large-scale video services. Past multi-path solutions that send duplicate copies of data may work for audio services [22], but they do not work for videos as the amount of traffic is much larger.

It is surprising that, to date, the feasibility and benefits of using multi-path in large-scale video services remain unclear. Therefore, in this paper, we present our large-scale experimental study of XLINK. *The key idea behind XLINK is to take the opportunity of QUIC as a user-space protocol and directly capture video QoE intent to control multi-path scheduling and management.* Specifically:

- Knowing the unpredictable nature of wireless links, we do not pursue accurate predictions of network characteristics. Instead, we rely on the client's QoE feedback to dynamically control packet re-injection's aggressiveness in the server's scheduler. XLINK's scheduler re-injects unacknowledged packets only when otherwise multi-path HoL blocking would impair the user perceived QoE, achieving performance and cost-efficiency at the same time.
- XLINK is designed to work with multiple concurrent QUIC streams. As a video may be downloaded through concurrent streams with each requesting a portion of it, stream blocking can happen when late-stream packets are queued before the re-injected copies of early-stream packets. With stream priority-based re-injections, XLINK carefully determines the sending order based on the urgency of the streams, enabling smoother streaming experience.
- XLINK is optimized for short videos. It introduces first-video-frame acceleration with video-frame priority-based re-injection, which allows video applications to further differentiate video frame importance beyond a stream granularity to avoid the excessive delay of in-flight packets on the slow path at video start-up.
- XLINK handles large path delay differences in heterogeneous networks with QoE-aware path management. First, it introduces wireless-aware primary path [3] selection, which takes care of different wireless technologies' path delays in the multi-path initialization process, providing quicker connection setup. Second, unlike MPTCP, multi-path ACKs (ACK_MPs) in XLINK are not restricted to the same subflow and XLINK is flexible in choosing the return path of ACK_MPs, which boosts the performance further.

In doing so, XLINK achieves superior performance compared to past approaches in terms of video start-up delays, video re-buffering rate, mobility support and, CDN traffic overhead. Moreover, as security and privacy are at the core of QUIC's design philosophy, there should be a basic premise that the multi-path QUIC keeps at least the same security level as QUIC and introduces no additional concerns. We design XLINK as a lightweight extension of QUIC while achieving the desired functionality and flexibility.

Contributions: We make the following contributions:

- We present the first large-scale experimental study of multi-path QUIC video transport in production environments to demonstrate the feasibility and deployability.
- We point out that the key to fundamentally addressing the above challenges is to leverage QUIC as a user-space protocol, allowing it to closely interact with an application and use video QoE for scheduling and path management.
- We reveal practical challenges that were less discussed in previous literature regarding performance, cost, mobility, compatibility, and network heterogeneity and share our experience in dealing with those challenges.

Main results: We conducted large-scale A/B tests of XLINK versus single-path QUIC with over 100K participants who voluntarily upgraded to a test-version of Taobao Android app that ran with XLINK's client, and we deployed XLINK's server with Taobao short-video service. Our data set consisted of over 3-million video plays. We observed that XLINK achieved 19 to 50% improvement in the 99-th percentile video-chunk request completion time, 32% improvement in the 99-th percentile first-video-frame latency, 23 to 67% improvement in the re-buffering rate at the expense of 2.1% redundant traffic. Therefore, we believe XLINK is a key step towards the widespread adoption of multi-path QUIC on the public Internet.

We would like to note that the innovation behind XLINK's utilization of remote feedback to control multi-path packet re-injection extends beyond the scope of end-to-end video delivery and can serve as a general high-performance multi-path transport mechanism. The stream and frame priority-based scheduling, on the other hand, leverage QUIC's ability to express video awareness, and hence are more QUIC-specific.

With the proliferation of video applications and wireless technologies, the traditional one-size-fits-all approach that optimizes video and transport at independent layers will no longer satisfy the ever-growing diversity of the user-perceived QoE. XLINK, which leverages the user-space nature of QUIC, pioneers innovations on a closer collaboration between video and wireless. The implications of XLINK's synergy of multi-path and the application's QoE-awareness extend beyond short videos and pave the new way for other exciting research areas such as live streaming, 360-degree videos, augmented reality (AR), and virtual reality (VR).

[3] Primary path is the path used to start a connection.

Claim: *This work does not raise any ethical issues. The research was conducted according to the ethical guidelines of ACM [23] and all reported data were desensitized and approved by our institution's review board.*

2 MOTIVATION

The development of XLINK is motivated by the following trends:

Short-form video explosion: Recent years have witnessed the explosion of short-form videos driven by apps such as TikTok, Reels, and Twitter [24]. E-commerce companies like Alibaba, Amazon, Ebay, and Redfin [1, 25, 26] are also shifting to short videos to showcase online products in mobile apps. Short videos impose more stringent QoE requirements because viewers are less tolerant of QoE impairments for short videos than long videos [27]. Meanwhile, consumers' appetites for video contents are moving towards 4K and beyond (e.g., AR and VR), which can request a bit-rate more than 85Mbps [28]. These changes urge mobile devices to overcome the spectrum shortage and link instability in wireless connectivity. XLINK offers an easy-to-deploy solution that enables billions of today's smartphones to get fast, smooth, and robust wireless connectivity.

Road to QUIC: QUIC was initially developed internally at Google to replace TCP [10]. Compared to TCP, QUIC is faster, more secure, and offers protection against protocol ossification. It is reported that QUIC now accounts for more than 40% of Google's [29] and 75% of Facebook's Internet traffic [30]. The increasingly widespread adoption of QUIC drives XLINK. We also learn from past pains and gains from the deployment of MPTCP. We show that the user-space property of QUIC is the key to overcome major hurdles such as unsatisfactory performance, difficulties in dealing with load balancers, obtaining OS-level support, and traversing middleboxes that block the use of MPTCP [5, 31, 32].

Better mobility support: Mobility support is vital in wireless communication. Unfortunately, today, roamings from Wi-Fi to cellular are either slow or prone to failures [33]. QUIC introduces connection migration (CM)[34], but CM requires resetting the congestion window after migration, which may not be suitable for video streaming, which needs sustained high bandwidth. Apple has shown the benefits of MPTCP to support Wi-Fi-LTE roaming in Siri, but to date, it is not clear whether multi-path remains effective in deployed video services. We develop XLINK to answer these questions and to explore the benefits of swiftly distributing packets according to link variations in high mobility scenarios.

Multi-path in 5G: The advent of 5G makes multi-path capabilities even more interesting. Although 5G offers higher bandwidth to fulfill data-rate needs, the smaller signal coverage due to more propagation loss and weaker penetration compared to LTE [35, 36] brings new challenges for 5G to meet its stringent reliability and QoS guarantees. On the other hand, Wi-Fi 6 will probably remain the most efficient method for indoor communication [37]. As a result, 3GPP introduced ATSSS in Release16, which features simultaneous usage of 5G and other non-3GPP access (e.g., Wi-Fi) [38]. Through a formal liaison, 3GPP has recently expressed interest to IETF for protocols that enable steering, switching, and splitting of traffic (primarily UDP) across multiple access where QUIC is a focal point [39]. XLINK keeps pace with such a trend with its ability to support 5G and Wi-Fi simultaneously.

3 EXPERIENCE WITH VANILLA MULTI-PATH QUIC

In this section, we present our experience with the vanilla multi-path QUIC (vanilla-MP [4]). Two significant challenges of multi-path performance are mobility and path delay difference. We first study the dynamics of vanilla-MP in mobile environments. Then we discuss the measured path delay difference when accessing our video servers via different wireless technologies. Finally, we show how vanilla-MP performs against single-path QUIC (SP) in a large-scale A/B test in our production environments.

3.1 Fast changing wireless links

To understand how vanilla-MP behaves in the mobile environment, we plot the dynamics of its in-flight packets replayed with a pair of Wi-Fi & LTE traces collected when walking on our campus shown in Fig. 1a and 1b with the Mahimahi emulation tool [40] [5]. The LTE trace was relatively stable, but on the contrary, the Wi-Fi trace changed rapidly, with its throughput dropping to almost zero from 1.7s to 2.2s; the congestion window (CWND) could not follow such rapid change. As a result, the scheduler still kept sending packets on that path, causing the number of in-flight packets to even go up at around 1.8s. Such a rapid variation could lead to severe HoL blocking since the video frame could not be delivered to the application until all the stagnant packets on the slow path (Wi-Fi) were recovered after a long period.

3.2 Path delays in heterogeneous networks

To understand the path delay differences. We measured RTTs when accessing our video services via different wireless technologies. We also deployed our own 5G SA testbed to understand 5G ultra-low latency [6]. We found that wireless technologies had a significant impact on path delays. The median path delay of LTE was 2.7 times and 5.5 times that of Wi-Fi and 5G SA, respectively, while the $90th$ percentile path delay of LTE was 3.3 times that of Wi-Fi. The path delay difference further increased with cross-ISP delays in multi-path [7]. The large differences in path delays could impact video start-up delays and request completion time in short video services.

3.3 Deployment of vanilla-MP

Table 1: Reduction of rebuffer rate (vanilla-MP vs. SP)

Days #	1	2	3	4	5	6	7
Improv. (%)	-34.6	-54.6	-47.7	-96.5	-77.8	-41.5	-48.3

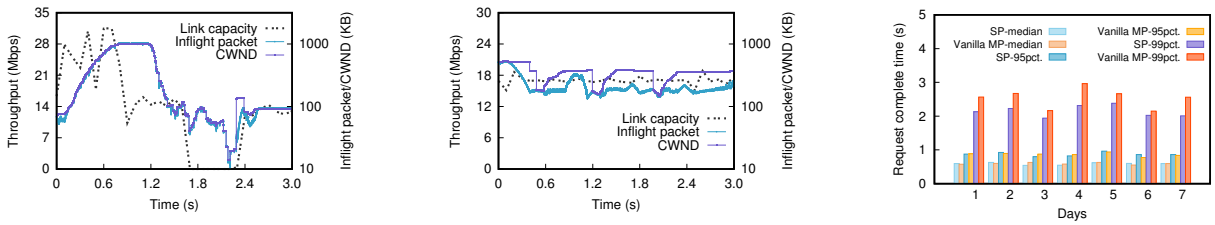
Finally, we verified the effectiveness of vanilla-MP by conducting a large-scale A/B test against single-path QUIC (SP) in our short video services. The experiment methods are discussed in Sec. 7.2. In Fig. 1c, we plot the median, $90th$ percentile and $99th$ percentile

[4] We implemented vanilla-MP with the min-RTT packet scheduler as described in MPQUIC [12]. The min-RTT packet scheduler is also the default packet scheduler used in Linux kernel MPTCP.

[5] We used the multi-path extension of Mahimahi. The experimental methods are discussed in Appx. B.

[6] At the time of writing, 5G SA is not commercially used.

[7] In practice, we also need to account the cross-ISP delay, which cause further increase in the delay of the secondary path. The relative increase of the measured cross-ISP LTE path delays in percentage are shown Table 4 in Appx. A and note that when employed as the secondary path, the delay could go up by 50% as the result of crossing ISP boarders.



(a) Wi-Fi Path (b) LTE Path (c) A/B Test (Vanilla-MP vs. SP)

Figure 1: Experience with vanilla multi-path QUIC (vanilla-MP). (a) and (b) vanilla-MP in fast varying wireless environments. (c) A/B test (vanilla-MP vs. single-path QUIC) on request completion time.

video chunk request completion time (RCT) collected throughout one week. The figure reveals the following findings: (1) Vanilla-MP was only effective at times on the median and 90th percentile RCT, and could lead to worse performance (days 1, 3, 4, and 5). The largest median RCT degradation was 16%. (2) Vanilla-MP always lead to degraded 99th percentile RCT, which could be even 28% worse than SP (days 4 and 7). In Table 1, we report the reduction of client-side video rebuffer rate (measured as the total amount of video rebuffer time normalized by the total amount of video play time) observed through the course of a week. A negative number indicated that the rebuffer rate of vanilla-MP was worse than that of SP. The rebuffer rate of vanilla-MP deteriorated. Instead of decreasing, it increased more than 34%, with the largest increase up to 96%. Such a result was not surprising due to the issues discussed above and therefore, vanilla-MP failed to meet the criteria of achieving no worse performance than single-path transport.

4 XLINK DESIGN OVERVIEW

In this section, we present the design overview of XLINK. The goal is to achieve optimal user-perceived QoE (e.g., low latency and small re-buffering) with the least possible overhead cost. As shown in Fig. 2, XLINK is designed as a lightweight end-to-end multi-path QUIC extension deployed in mobile apps and edge servers. It enables a multi-homed mobile client to communicate to a remote server with simultaneous transmissions over multiple wireless interfaces (e.g., Wi-Fi and cellular). Unlike past solutions such as MPQUIC and MPTCP that operate unassisted from applications, XLINK is driven by the recent trends of cross-layer network designs [41] and closely integrates transport with video apps to achieve high performance and cost-efficiency at the same time. The core of XLINK is to take the opportunity of QUIC as a user-space protocol and leverage the user-perceived video QoE in multi-path scheduling and management.

Architecturally, XLINK's QoE-driven scheduling is built on top of a client-server feedback mechanism. A XLINK client captures user-perceived QoE signals (e.g., video player cached frames and video player frame-rate) and uses ACK_MP extension frame (Sec. 6 and Fig. 16) to carry those signals to a remote video server to control its scheduling. The use of QoE_control_signal field controls the coupling and decoupling of multiple paths. It allows XLINK to overcome multi-path HoL blocking without incurring unnecessary cost overhead, which is crucial for large-scale deployability (Sec. 5.2). XLINK further handles large path delay differences by offering first-video-frame acceleration (5.1), wireless-aware primary path selection (5.3), and fastest-path ACK_MP to avoid excessive delay from the slow path and improve video start-up.

Algorithmically, XLINK utilizes packet re-injection to decouple multiple paths. Unlike past re-injection [6], XLINK implements priority-based re-injection at two levels: transport (QUIC stream) level and application (video frame) level (Sec. 5.1). The stream priority-based re-injection accounts for QUIC's concurrent streams that request different portions of a video. It ensures that re-injected packets of an early stream are sent before later streams' packets, thus preventing stream blocking at transport. The video-frame priority-based re-injection differentiates video frame urgency within a stream. It treats the first frame of a video with the highest priority to speed up video start-up. In terms of QoE feedback control, XLINK introduces double thresholding control (5.2.2), which achieves responsiveness and cost efficiency and offers flexibility to balance performance and cost.

At the protocol level, XLINK builds on top of the multi-path extensions proposed in draft [11], which incorporates PATH_STATUS and ACK_MP extension frames to manage path status and acknowledge packets received from different paths. The only difference is that the current XLINK implementation (used in this experiment) sends QoE feedback as an additional field in ACK_MP frame, instead of sending the QoE feedback in the independent QOE_CONTROL_SIGNALS frame specified in the draft.

5 QOE-DRIVEN SCHEDULING AND PATH MANAGEMENT

This section discusses the details of QoE-driven multi-path scheduling and path management, which enables XLINK to achieve superior user-perceived QoE with minimized cost overhead. It consists of three major components: stream and video-frame priority-based packet re-injections, QoE feedback control, and QoE-aware path management. We overcome multi-path HoL blocking, stream blocking, and excessive delay at video start-up with stream and video-frame priority-based re-injections. To reduce cost, QoE feedback is used to control the redundancy associated with re-injection. We further handle path delay differences in heterogeneous networks with QoE-aware path management that incorporates wireless-aware primary path selection and fastest-path multi-path ACK.

5.1 Priority-based re-injection

Why re-injection? Re-injection is used to decouple multiple paths. As discussed earlier, the root cause of multi-path HoL blocking is the coupling of multiple paths when the scheduler splits packets

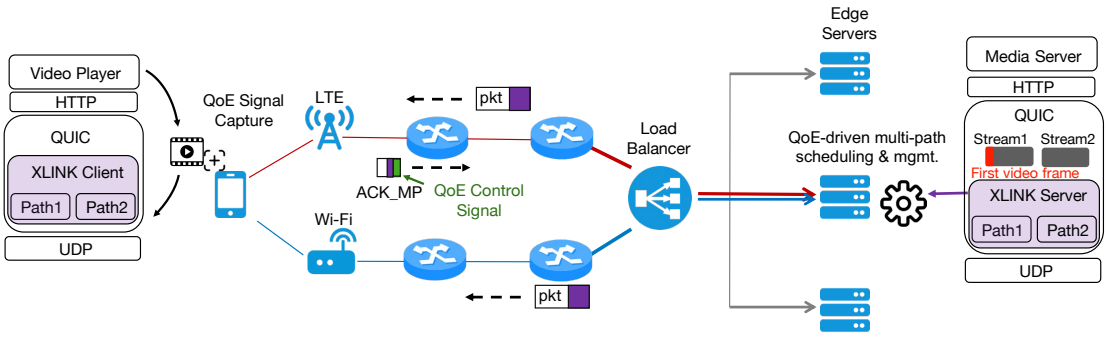


Figure 2: The overview end-to-end architecture of XLINK.

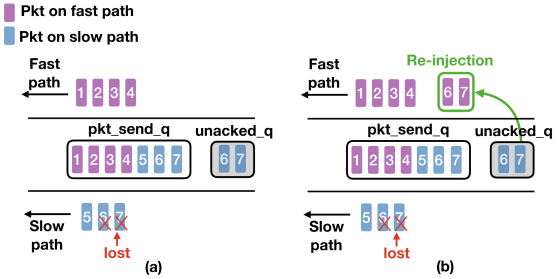


Figure 3: Use re-injection to overcome multi-path HoL blocking: (a) Without re-injection, packets lost on the slow path would block the fast path. (b) With re-injection, lost packets on the slow path can be quickly recovered from the fast path.

across them. To explain how multi-path HoL blocking happens through such coupling, Fig. 3(a) shows the typical process of a scheduler splitting packets from its sending queue (pkt_send_q) across a fast path (purple) and a slow path (blue). The packets sent by the fast path and the slow path complement each other and the client waits successful delivery on both paths to obtain the whole chunk of video. Blocking happens if pkt 6 and pkt 7 on the slow path are lost (maybe due to a sudden link outage) because the client cannot proceed until the loss recovery on the slow path, which can take as long as a retransmission timeout (RTO). Re-injection is a technique that allows us to decouple multiple paths with the use of redundant duplicate packets, which is shown in Fig. 3(b). In re-injection, the sender keeps track of a queue for unacknowledged packets (unacked_q). Back to the same example, when there are no more packets in the pkt_send_q to send, the sender can send duplicates of the unacknowledged packets 6 and 7 with re-injection into the fast path without waiting for the loss recovery on the slow path, allowing the receiver to continue consuming data without suffering from the blocking effect.

Priority-based re-injection. However, traditional packet reinjection is not enough to achieve good video QoE. The first thing we need to address is the stream blocking effect. Unlike TCP, QUIC transport layer has the concept of *QUIC Stream*. A connection can carry multiple streams with each separately flow controlled and loss recovered. In short-video transport, the video player may simultaneously request multiple streams, with each downloading a small portion of the video [8]. As shown in Fig. 4 (a) and (b), the pkt_send_q now contains two streams, Stream 1 and Stream 2. The

[8]When the network is good, the use of multiple concurrent streams allows the media player to pre-fetch video chunks.

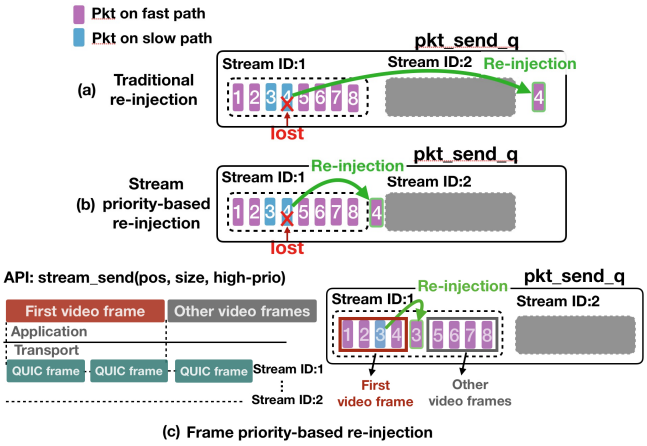


Figure 4: Different modes of re-injection: (a) Traditional (appending) mode, (b) stream priority-based mode to address stream blocking and (c) video-frame priority-based mode to address video frame blocking.

traditional re-injection works in an appending mode in Fig. 4(a). If pkt 4 is lost on the slow path in this mode, the scheduler can only re-inject it at the end of the pkt_send_q, behind Stream 2, which is not optimal. As the contents of streams play in sequence, stream 2 now blocks the delivery of stream 1. Indeed, an early stream should enjoy high priority re-injection so that the pkt 4 is re-injected right after Stream 1. In XLINK, we employ stream priority-based re-injection (Fig. 4(b)) to take care of concurrent QUIC streams. In this mode, when the sender sends out the last packet in Stream 1, it immediately checks the unacked_q to look for packets of the same stream priority. If any, it inserts those packets before the unsent packets of lower priority streams, as shown in Fig. 4(b), thus preventing stream blocking.

First-video-frame acceleration. XLINK further introduces video-frame priority-based re-injection to accelerate first-video-frame delivery, which is critical for short videos. Without it, the use of multi-path in the presence of a large path delay difference may suffer from a slow video start caused by the video frame blocking effect. The reason is that a multi-path scheduler may put a first-video-frame packet on an ill-conditioned path if the congestion window of a well-conditioned path is full (e.g., pkt 3 in blue in Fig. 4(c)). In this case, the stream-level granularity offered by stream priority-based re-injection is not enough because the re-injected packet still has to wait for other video frames in the same stream to be delivered (See.

Fig. 4(b)). XLINK addresses this problem with video-frame priority-based re-injection shown in Fig. 4(c). In this mode, XLINK provides *stream_send* API to the application to express video QoE-awareness at a finer granularity. Specifically, to accelerate the first video frame, the application can set the stream data containing the first video frame at the highest priority with position and size parameters that indicate the video frame's relative location. When enabled, XLINK checks the unacknowledged packet from the first video frame (pkt 3) after sending out the last first-frame packet (pkt 4). If there is any, the scheduler re-injects it (pkt 3) before any unsent packets of the other video frames in the same stream. The re-injected copy can go through the fast path this time, which may arrive earlier than the original packet. Therefore, the video-frame priority-based re-injection avoids the slow path's excessive delay and significantly improves the video start-up speeds.

5.2 QoE feedback and re-injection control

The problem with packet re-injection is that it, unfortunately, introduces a lot of redundant packets. **In our case, we found that direct applying of re-injection increased the total amount of traffic by 15%.** The humongous cost overhead [9] would be unacceptable for large-scale deployment. Besides the cost problem, redundancy could also impact the overall throughput. In order to address this challenge, XLINK leverages the client's QoE feedback to control the cost overhead associated with packet re-injection while ensuring user-perceived QoE.

Indeed, redundant packets are not always needed as the video player caches video chunks. If the buffer occupancy level is high, the play-time left until the next possible re-buffering is long, and hence, the urgency of using re-injection is low. On the contrary, if the buffer occupancy level is low, the time left until the next possible re-buffering is short and, hence, the urgency of using re-injection is high. Knowing that the client video player's buffer occupancy is a strong indicator of the user-perceived QoE, XLINK captures buffer occupancy information and sends it back to the server to control its re-injection usage. These signals are conveyed in the QoE_Control_Signal field of the ACK_MP frame in Fig. 16. The definition of QoE signals can be flexible. In our case, we capture four types of signals from the client's video player: (1) the number of cached bytes (*cached_bytes*), (2) the number cached of frames (*cached_frames*), (3) the bitrate (*bps*), and (4) the framerate (fps).

5.2.1 ***Capturing QoE signals.*** The process of how XLINK captures QoE feedback signals is illustrated in Fig. 5. Here we focus on the video pipeline, which consists of the Media Source, the Source Pipe, the Decoder, the Decoder Pipe, and the Renderer. The MediaCacheService responds to video play request from media player and sends out HTTP range requests to server to obtain video chunks. TNET is an Android network SDK used in Taobao client, which delivers the QoE signals to XLINK. The incoming media data is first processed by the Media Source where the audio and video frames are split and cached in the Source Pipes, which subsequently send the frames to their respective Decoders for the actual decoding. The Source Pipe keeps track of the number of cached frames and the number of cached bytes. The Decoder has the knowledge of

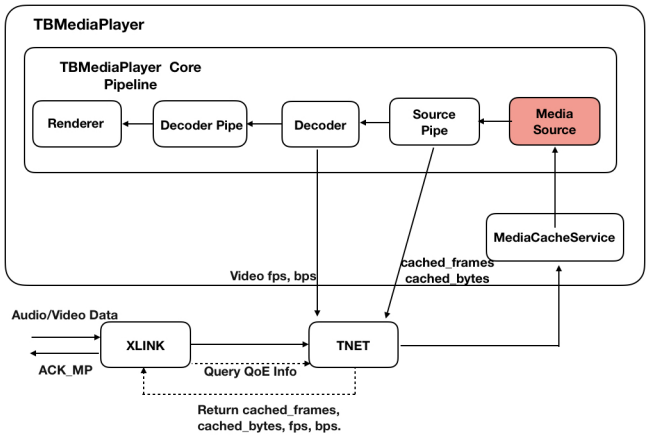


Figure 5: Illustration of how XLINK captures QoE feedback signals from the media player.

the frame-rate and bit-rate. In order to obtain the QoE information, the Source Pipe and Decoder keep sending the these updated metrics to TNET, periodically. When XLINK wants to send out a QoE feedback, it queries the TNET. If the QoE information is updated in TNET, TNET responds to XLINK's query, then XLINK encapsulates the QoE feedback information into the QoE Control Signal field in ACK_MP frames as shown in Appx. C.

5.2.2 ***Double thresholding control.*** The algorithm that controls the re-injection usage needs to satisfy three properties: responsiveness, cost-efficiency, and flexibility. (1) It must be responsive enough when re-injection is urgently required. (2) It should accurately prevent any unnecessary re-injections. (3) It needs to offer flexibility to adjust the balance between performance and costs. We introduce double thresholding control to meet the above needs. The basic form of the algorithm is shown in Alg. 1. The inputs of this algorithm are the four types of QoE signals as described above, and the output of the algorithm is the decision of whether re-injection should be enabled. At a high level, the algorithm can be divided into three steps:

Step1: Computing play-time left. We first estimate the video play-time left Δt in the client's buffer with the QoE feedback. One could use quotient of the $cached\_frames$ divided by fps or the quotient of *cached_bytes* divided by bps to compute Δt. When the video is not encoded with constant bitrate and the framerate is high, we recommend computing Δt using $cached\_frames$ and fps since the *bps* could exhibit large variations. However, if the framerate is very low, computing based on *cached_bytes* and *bps* is desired. Basically, one should look at both the bit-rate and the frame-rate. This allows us to get a more conservative estimate [10].

Step 2: Double thresholding. The second step is double thresholding, in which we compare the play-time left Δt with two thresholds, T_{th1} and T_{th2}, where we set $T_{th1} < T_{th2}$. The two thresholds are applied for different purposes. The first threshold T_{th1} is used to ensure responsiveness. If $\Delta t < T_{th1}$, it means that the play-time left is too little and we need to turn on re-injection immediately, so Alg. 1 returns *true*. The second threshold T_{th2} is applied to obtain

[9] The traffic cost is $0.085 per GB[42]

[10] In the above discussion, we assume that the QoE feedback is frequent enough. However, in practice, the feedback frequency is determined by the video player. It should be noted that if the QoE feedback is not frequent enough, the video play-time left Δt needs to be extrapolated.

cost-efficiency. If $\Delta t > T_{th2}$, it means that the cached data on the client is sufficient, and we can safely turn off re-injection to reduce cost, so Alg. 1 returns *false*. The combination of two thresholds offers flexibility as one can easily tune these thresholds to trade performance with cost.

Step 3: Comparing with delivery time. When Δt is in the range of $[T_{th1}, T_{th2}]$, the buffer occupancy has a medium level, so the delivery time of in-flight packets plays a role in the decision. We further compare Δt with the estimated maximum delivery time of in-flight packets $deliverTime_{max}$. If $\Delta t < deliverTime_{max}$, one of the paths may be too slow, so the re-injection should be turned on to accelerate the packet delivery using the other path. If $\Delta t > deliverTime_{max}$, the in-flight packets will arrive in time, so the re-injection should be turned off to save cost. $deliverTime_{max}$ is calculated as the maximum RTT plus its variation δ of all paths that have unacknowledged packets, as shown below:

$$deliverTime_{max} = \max_{p \in \mathbf{P} \wedge unacked\_q_p \neq \varnothing} \{RTT_p + \delta_p\} \quad (1)$$

Example. To illustrate how Alg. 1 overcomes HoL blocking with reduced cost in fast-changing wireless environments, we plot the dynamics of the client's buffer level and the amount of re-injected packets vs, time in one example as shown in Fig. 6. We test vanilla-MP (Fig. 6b), re-injection without QoE control (Fig. 6c) and re-injection with QoE control (Fig. 6d) replayed with the same network traces shown in Fig. 6a. We can see that vanilla-MP, whose buffer level drops to zero several times as Path 1 deteriorates, suffers from severe multi-path HoL blocking and results in frequent video re-buffering in Fig. 6b. Re-injection is effective in overcoming multi-path HoL blocking, so when path 1 deteriorates, Fig. 6c and Fig. 6d can maintain sufficient cached bytes in their buffer. However, without QoE control, Fig. 6c uses re-injection recklessly as it re-injects packets even when the buffer level is high, causing unnecessary, redundant traffic. With the help of QoE control, Fig. 6d only uses re-injection when the buffer level is low, so it avoids any unnecessary usage of re-injection when the buffer level is high. As a result, Fig. 6d is able to ensure the smoothness of the video play with the least traffic overhead [11].

Performance and cost tradeoffs. The two thresholds T_{th2} and T_{th2} offer flexibility in the trade-off between performance and cost. For example, a larger T_{th1} provides better tail performance at the cost of increasing the lower bound of traffic overhead C_{min}. While T_{th2} allows us to control the upper bound of traffic overhead C_{max} and we have, $C_{min} >= \beta * Prob(\Delta t < T_{th1})$ and $C_{max} <= \beta * Prob(\Delta t < T_{th2})$, where β [12] is the cost overhead with re-injection turned on, which can be roughly estimated as 15% in our case. In real world deployment, we recommend that T_{th2} and T_{th2} are set according to the distribution of the client video buffer occupancy.

5.3 QoE-aware path management

This section describes how XLINK's QoE-aware path management handles path delay differences in heterogeneous networks. We first introduce the wireless-aware primary path selection in the multi-path initialization and then discuss the multi-path ACK path selection strategies.

Algorithm 1: Double Thresholding Control

Input : T_{th1} -the first threshold; T_{th2} - the second threshold; $\boldsymbol{P}$ - all available paths; Latest QoE feedback: cached_bytes, cached_frames, bps and fps.

Output : re-injection decision

```
Δt ← est(cached_bytes/bps, cached_frames/fps)
if Δt > T_th2 then
    return false
if Δt < T_th1 then
    return true
maxDeliverTime ← 0
foreach p ∈P do
    if exist_no_unack_pkts(p) then
        continue
    deliverTime ← RTT_p + δ_p
    if deliverTime > maxDeliverTime then
        maxDeliverTime ← deliverTime
if Δt < maxDeliverTime then
    return true
return false
```

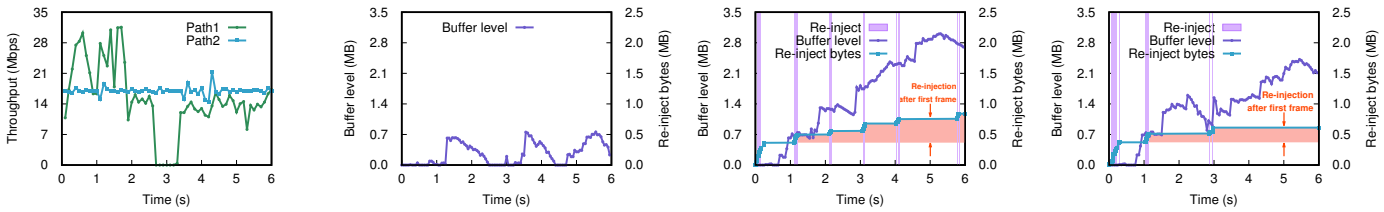
Wireless-aware primary path selection. XLINK carefully determines the primary path to initialize a connection based on the types of wireless interfaces. Past studies have shown that due to path delay differences, primary path selection impacted the performance of MPTCP [43]. The path delay difference is further enlarged with 5G SA's advent because: (1) Unlike 5G NSA that shares the same core network with LTE, 5G SA employs a new and independent core network. (2) 5G SA natively supports edge computing and brings content delivery services further close to the access networks' edge. To accommodate the upcoming 5G SA, we make XLINK's primary path selection wireless-aware. For example, we can set the following order: *5G SA > 5G NSA > WiFi > LTE* [13]. We measured the first-frame delivery time vs. different frame sizes when starting a multi-path connection from different wireless interfaces in Fig. 7. The measurement was conducted with our 5G SA test-bed and enterprise Wi-Fi. The influence of primary path selection on first-video-frame delivery time is significant. Simply starting with the right primary path can offer a much faster video start-up.

Fastest-path Multi-path ACK. Finally, but importantly, XLINK utilizes fastest-path Multi-path ACK. Different from MPTCP, whose ACK is supposed returned on the same sub-flow (original path) [4]. XLINK allows ACK_MP returned from any of the paths, which gives more flexibility. There are two basic strategies: ACK_MP on the fastest path (min-RTT path), and ACK_MP on the original path. In XLINK, we use the fastest path to transmit ACK_MP. We show the effect of the two ACK path selection strategies with Cubic congestion control in Fig. 8. We measured the request completion time

[11] Note that in this example, we focus on the dynamics of the video start-up phase. As time goes on, the redundancy ratio of re-injection w. QoE control drops significantly.

[12] For simplicity, β is expressed as a constant, but in reality, the value of β can change.

[13] The ranking is subject to change in different countries and states. One should follow local statistics.



(a) Network traces (b) Vanilla-MP (c) Re-injection wo. QoE (d) Re-injection w. QoE

Figure 6: Example of how Alg. 1 overcomes MP-HoL blocking with reduced cost overhead in fast changing wireless environments: (a) Path traces used in this experiment. (b-d) Dynamics of client's buffer occupancy level (cached bytes) and server's re-injection packets (in bytes) for (b) vanilla-MP, (c) re-injection without QoE control, and (d) re-injection with QoE control.

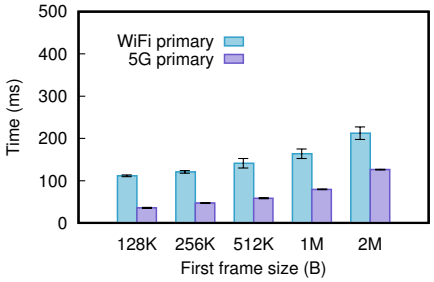


Figure 7: First-video-frame delivery time when starting a connection from a 5G or a Wi-Fi interface.

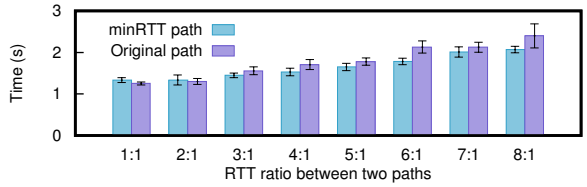


Figure 8: ACK_MP path selection strategies with Cubic congestion control: shortest-path (min-RTT) vs. original path.

of a 4MB load by changing the path RTT ratio between two paths of equal bandwidth with Mahimahi. As the delay ratio becomes larger, ACK_MP on the fastest path begins to show advantages. The reason is that for Cubic, faster ACK return helps the congestion window grow faster, which yields better throughput.

6 PROTOCOL AND IMPLEMENTATION

We describe the protocol and implementation of XLINK in this section. We start with how XLINK extends QUIC to multi-path, followed by the integration of XLINK into android apps and video services. XLINK is implemented based on our multi-path QUIC draft [11], which extends IETF QUIC to multi-path with smallest possible modifications. Different from past proposals [7] that heavily relied on the "uni-flow" concept, we extend multi-path on top of the concept of the bidirectional paths, which readily fits into the nature of both cellular and wifi links that cover the majority of multi-path applications in QUIC while keeping the design simple and easy to implement. In doing so, we are able to re-use most of the current QUIC transport design with the sole addition of three new frames. More importantly, our design supports QoE feedback that is needed to enable XLINK's feedback-based dynamic scheduling.

The key design points are: (1) Different paths are identified by the sequence number of connection IDs (CIDs). To make packet loss detection and recovery convenient, we use separate packet number space for each path. (2) We keep QUIC packet header formats unchanged to avoid the risk of packet being blocked by middle-boxes. (3) All paths that belong to one connection share the same encryption key, but we incorporate a mechanism to enable each path to obtain a unique nonce in AEAD. (4) We incorporate PATH_STATUS and ACK_MP extension frames to support multi-path functionality and QoE feedback mechanism.

Multi-path initialization. The multi-path path initialization process is shown in Fig. 9. XLINK first initializes the primary path in the same way that single-path QUIC does, except that during the first handshake, the client includes a *enable_multipath* transport parameter. If the server replies with a *enable_multipath* parameter, then both end-hosts know that multi-path is supported. If not, they fall back to single-path QUIC. Before initializing a new path, the client needs to provide at least one unused available CID (e.g. C1 with sequence number 1), and the server needs to provide at least one unused available CID[14]. In order to setup a new path, the client chooses an available Connection ID S2 as the Destination Connection ID in the new path. The exchange of CID is done with the *NEW_CONNECTION_ID* frame defined in QUIC [34]. To avoid path spoofing attack, XLINK uses *PATH_CHALLENGE* and *PATH_RESPONSE* frames defined in [34]. Once multi-paths are initialized, XLINK uses ACK_MP frame instead of ACK frame to send acknowledgement.

Frame extension. We use the PATH_STATUS frame and ACK_MP frame to support multi-path functionality and QoE feedback. The PATH_STATUS frame is used to help manage multi-paths, which informs the peer of the current status of a path, and the peer should send packets according to the preference expressed in these frames. Available values of PATH_STATUS are *Abandon(0)*, *Standby(1)*, and *Available(2)*. Endpoints use the sequence number of the CID used by the peer for PATH_STATUS frames (describing the sender's path identifier). The ACK_MP frame allows for convenient loss

[14]When client wants to start a new path, it checks whether there are unused available CIDs on each side, and chooses an available CID S2 as the DCID in the new path.

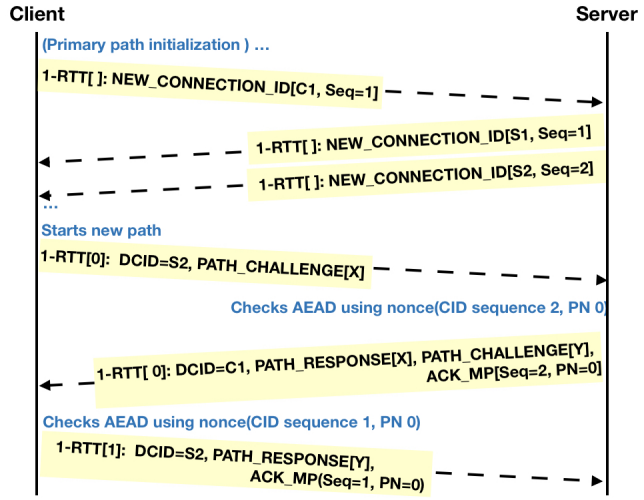


Figure 9: XLINK's path initialization process

detection and recovery on each path by incorporating a path index field (CID sequence number). In our experiments, it also supports QoE feedback between a client and a server by incorporating the *QoE_Control_Signal* field. The structure of ACK_MP is shown Appx. C. In our draft [11], we further introduce the QOE_CONTROL_SIGNAL frame extension, which allows us to send the QoE feedback independently without being restricted by ACK frequency.

Path close. Both client and server can close a path, by sending PATH_STATUS frame which abandons the path with a corresponding Path Identifier. Once a path is marked as "abandon", it means that the resources related to the path can be released. In scenarios such as client detects the network environment change (client's 4G/Wi-Fi is turned off, Wi-Fi signal is fading to a threshold), or endpoints detect that the quality of RTT or loss rate is becoming worse, client or server can terminate a path immediately.

Packet protection. The general principles of packet protection are not changed for QUIC Multipath. No changes are needed for setting packet protection keys, initial secrets, header protection, use of 0-RTT keys [11]. However, the use of multiple number spaces for 1-RTT packets requires changes in AEAD usage. For QUIC multipath, the construction of the nonce starts with the construction of a 96 bit path-and-packet-number, composed of the 32 bit Connection ID Sequence Number in byte order, two zero bits, and the 62 bits of the reconstructed QUIC packet number in network byte order. If the IV is larger than 96 bits, path-and-packet-number is left-padded with zeros to the size of the IV. The exclusive OR of the padded packet number and the IV forms the AEAD nonce.

Work with Load Balancers. XLINK works with load balancers that implement the routing algorithm specified in QUIC-LB draft [44]. We use consistent hashing on Connection IDs in load balancers to ensure that multiple paths are routed to the same real servers. In order to do so, a real server encodes a server ID in the CID issued to the client.

Integration with android apps. A XLINK client, written in C, is implemented in XQUIC [45], which is the Alibaba's implementation of the IETF QUIC, and is integrated into Taobao Android App. XLINK offers APIs for applications such as video players to pass information (e.g., cached bytes, cached frames, current bit-rate, and frame-rate) to QUIC. The test package can be released weekly on the client-side.

Deployment in CDN servers. A XLINK server is also written in C(also implemented in XQUIC [45]), and is deployed in a multi-process architecture CDN server. To deliver received packets to the right process that holds the context of a QUIC connection, we use consistent hashing on a process ID that is encoded in the reserved bytes in the CID. The algorithm parameters of XLINK are added as configuration items that can be updated in hours.

7 EVALUATION

In this section, we present the evaluation of XLINK, which consists of two parts: online evaluation and controlled evaluation.

Online evaluation: In this part, we report data from real users who upgraded to Taobao Android app with XLINK. We start by examining the change of client-side video player's buffer-level distribution and the corresponding redundant traffic cost vs. the choice of double thresholds. Then, we report the results from our large-scale A/B tests, where we conducted day-to-day comparisons between two contrast groups (single-path QUIC and XLINK) running in parallel. The total number of participants was over 100K and our measurements consisted of over 3-million video plays. In collaboration with a mobile carrier, the participants who used multi-path in the experiment enjoyed zero-rated cellular data [15]. Our A/B test results include both video request completion time and QoE metrics. In the QoE metric part, we focused on the rebuffer rate improvement and first frame delivery time improvement.

Controlled evaluation: In this part, we conduct measurements in controlled environments which allowed us to compare different methods with repeatable network conditions. We first conducted high mobility evaluations where we compared the performance of XLINK and other multi-path solutions with network traces collected in extreme mobility scenarios. Then, we measured energy consumption of using XLINK on cellphones when downloading video files of various sizes.

Unless otherwise specified, the congestion control algorithms used in experiments were Cubic. For vanilla-MP, MPTCP, and XLINK, we used "decoupled" congestion control, which was the typical configuration for mobile multi-path transport [46, 47]. We also discuss congestion control issues in Sec. 9.

7.1 Buffer-level and cost overhead vs. double thresholds.

We start by investigating the choice of double thresholds in the QoE control algorithm by changing the upper and lower thresholds in Alg. 1. To determine the threshold values, we first measured the play-time left distribution from QoE feedbacks [16] when the control was off. Then we chose thresholds $(th(X), th(Y))$, where $th(X)$ and $th(Y)$ denoted the X-th and Y-th percentile values in the distribution (e.g., $Prob[x > th(X)] = X\%$) [17]. Fig. 10 shows the

[15] Zero-rated data means the mobile carrier does not count the data used by a user for a specific app.

[16] We measured the buffer level after the video start-up phases.

[17] E.g, if X is 90, 90% of the play-time left values are greater than $th(X)$.

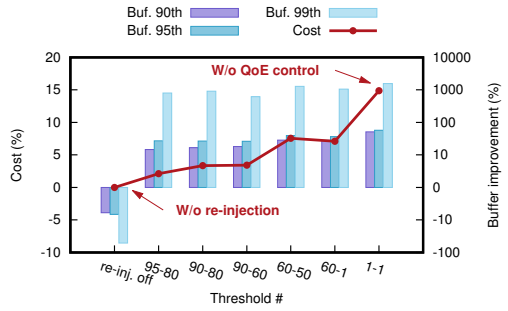


Figure 10: Client's buffer occupancy levels and traffic cost overhead vs. different threshold settings.

client's buffer level improvements over SP vs. threshold settings, where $(th(X),th(Y))$ is simplified as (X,Y). We also plot the cost associated with these threshold settings in Fig. 10. We observed the following:

- Re-injection was necessary. When re-injection was off, the buffer levels dropped significantly.
- QoE control was necessary. When it was off in the setting $(1,1)$, the traffic overhead reached 15%.
- The overhead cost was lowered bounded by $\beta(1-X)$ and upper bounded by $\beta(1-Y)$, where β can be approximated as 15%, which was expected (see Sec. 5.2.2).
- A moderate threshold value achieved good performance with small cost (e.g., (95, 80)). Further increasing the threshold saw diminished returns.
- Comparing delivery-time was useful in ruling out unnecessary re-injection cases to control cost when the upper threshold was large, as can be seen in $(90,80)$ vs. $(90,60)$ and also $(60,50)$ vs. $(60,1)$.

We further calculated the percentage of samples whose play-time left was smaller than 50ms, which is considered as a danger level that could lead to possible video rebuffer. The reduction of such percentage is shown in Table 2, where the threshold settings are the same as Fig. 10. The percentage reduction of the buffer levels smaller than 50ms correlated with the improvement of the $99th$ percentile buffer level and a lower-threshold at $th(95)$ was sufficient to cover the poor network conditions. Therefore, the most cost-efficient combination was (95, 80), which achieved 66% improvement in the re-buffering probability with an overhead cost of 2.1% (See Fig. 10). Further increasing the cost saw the same level of improvements, despite the daily variations due to the still limited experiment size. These results prove the effectiveness of Alg. 1 and show that we can balance performance and cost-efficiency with the proposed algorithm.

Table 2: Percentage reduction of buffer levels < 50ms

Thresh. #	95-80	90-80	90-60	60-50	60-1	1-1
Improv. (%)	66.14	57.59	55.65	78.87	64.74	62.23

7.2 Large-scale A/B test

Video request completion time. We report A/B test result of video request completion time over the course of two weeks. Fig. 11 shows the median, $95th$ and $99th$ percentile request completion

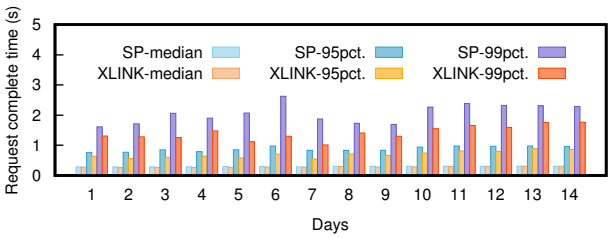


Figure 11: A/B test results (XLINK vs. SP) on request completion time.

time (RCT) of video chunks. As discussed earlier in Sec. 3, vanilla-MP underperformed SP at times. In contrast, XLINK consistently outperformed SP in both the median and the tail RCT because it was able to overcome multi-path HoL blocking with the QoE-driven multi-path scheduling and management. We observed 2.3% to 8.9%, 9.4% to 34% and, 19% to 50% day-to-day improvements in the median, 95th, and 99th percentile RCT, respectively [18]. The large improvement at high percentiles was contributed by more reliability and diversity gain of multi-paths at tail distribution.

QoE metric #1: video rebuffer rate. The reduction (improvement) of client-side video rebuffer rate observed through the course of one week is shown in Table 3. The video rebuffer rate, which measures video playback smoothness, is defined as the total amount of video rebuffer time normalized by the total amount of video play time. In other word, the video rebuffer rate is calculated as sum(rebuffer time)/sum(play time). Compared to SP QUIC, XLINK consistently outperformed SP in video rebuffer rate. The observed rebuffer rate reduction was significant, which ranged from 23.8% to 67.6%. Such a result was in accordance with the video request completion time shown in Fig. 11. The evaluation results showed that XLINK was effective in improving the quality of user-perceived experience in terms of video playback smoothness.

Table 3: Reduction of rebuffer rate (XLINK vs. SP)

Days #	1	2	3	4	5	6	7
Improv. (%)	27.00	48.41	55.16	66.36	67.67	23.81	53.99

QoE metric # 2: first-video-frame latency. We measured the effect of the first-video-frame acceleration by enabling and disabling this functionality during the experiment. The improvements of the first-video-frame latency over SP at different percentiles are plotted in Fig. 12. Without first-video-frame acceleration, the video latency became much worse than that of SP where the 99th percentile latency was even 14% slower. Such degradation was caused by the excessive delay introduced by the slow path. In contrast, first-video-frame acceleration avoided this excessive delay, and offered a much faster video start-up as its performance was lower-bounded by the fast path. The 99th percentile improvement was more than 32% than that of SP. Note that the improvement became larger towards the tail, which was expected because the difference between the fast and slow paths also became larger at tail.

7.3 Extreme mobility

Next, we investigate how XLINK performed in extreme mobility scenarios. We collected LTE and onboard Wi-Fi traces in subways

[18] The only degradation was the median RCT value in day 14, which dropped 0.7%.

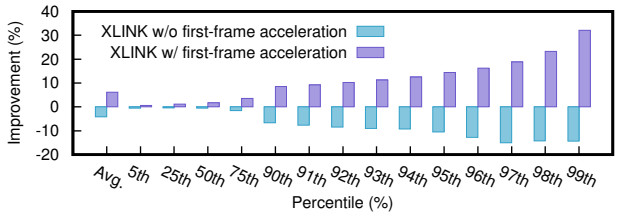


Figure 12: A/B test results (XLINK vs. SP) on first-video-frame latency with first-video-frame acceleration and without first-video-frame acceleration.

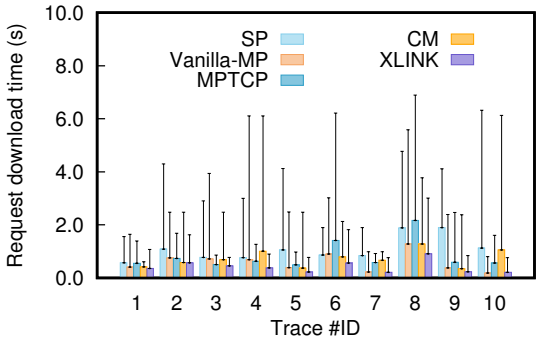


Figure 13: Extreme mobility experiment: request download time under different extreme mobility traces.

and high-speed rails and conducted trace-driven evaluation with Mahimahi emulation tools [40]. More details about our trace-driven evaluation and sample traces are listed in Appx. B. In comparison to XLINK, we also evaluated SP, vanilla-MP, MPTCP, and QUIC connection migration (CM).

Fig. 13 shows the video-chunk request completion time with median and max values, and we make the following observations: (1) SP without CM performed poorly with any of these traces due to the lack of mobility support. (2) CM showed improvement compared to SP as it could migrate to a new path when the old path was degraded. However, under extreme mobility, the new path was also likely to degrade immediately, so migration might not be effective and even lead to worse results than SP in some cases. Moreover, in CM, the *cwnd* needed to be reset after migration, which triggered slow start in congestion control, and it was the client's responsibility to probe a path to detect path degradation, which could take several round-trips. As a result, CM was not responsive enough when hand-off frequency became high. (3) MPTCP and vanilla-MP showed improvement over SP in some cases, but in the meantime, they also suffered from severe HoL blocking under high-speed link variations which caused performance degradation in many cases.

In contrast, XLINK consistently outperformed other methods with much smaller RCT in both median and max values. Because XLINK not only effectively aggregated wireless bandwidth but also swiftly adapted its packet distribution across fast varying links with the real-time QoE feedback control, it overcame fast link variations and frequent hand-offs, thus offering much better support in extreme mobility.

7.4 Energy consumption

It is essential to understand the energy consumption, as battery life affects users' willingness of using multi-path. We measured the

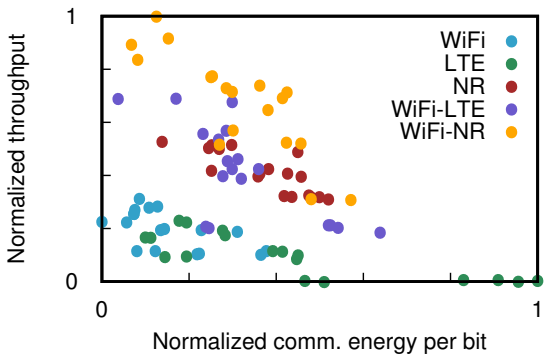


Figure 14: Normalized energy per bit and throughput (down-link) of XLINK installed on different Android models

normalized communication energy per bit vs. throughput of XLINK installed on three popular 5G-NSA-capable Android smartphone models and downloaded different sized loads (10MB-50MB).

We built an in-house energy monitoring tool with the open-source Android APIs [48], which logged the following information of communication modules (Wi-Fi and cellular) from the phone's OS kernel: time-stamp, instant current, voltage, WiFi RSSI, and cellular RSSI. To insulate the communication module energy measurements from other background noise, we first turned on the "airplane" mode to kill all background processes while keeping the same screen brightness. Then we started downloading files with XLINK with the APP running to record the energy information. The Android phone in our tests used Snapdragon 765G and Kirin 990 chipsets. As for 5G new radio (NR) use cases, we want to understand when the 5G throughput could not attain its peak-rate [19] such that multi-path should be enabled, so we capped each link's speed to 30Mbps.

The result is shown in Fig. 14 (the top left corner is better). We note that the use of double links could increase the instantaneous power, but the energy per bit is not necessarily high because the energy is equal to power × time, where the communication time reduces with higher throughput. We make the following observations:

- In terms of throughput, both Wi-Fi-LTE and Wi-Fi-NR showed significant improvements over their single-path counterparts.
- In terms of energy per bit, Wi-Fi-LTE and Wi-Fi-NR improved over LTE and NR, respectively. Wi-Fi was more energy-efficient, but its throughput was much lower than XLINK, so there is a trade-off and XLINK is more suitable for applications requiring high bandwidth and low latency (e.g., video applications).

8 RELATED WORK

Multi-path extensions over QUIC. QUIC, a user-space transport over UDP initiated from Google [49], has changed the landscape of web transport. Since 2016, the IETF QUIC working group has been working on the various parts of its specifications (e.g., transport, security, and recovery) [50], and they are going to become RFCs in 2021 [14]. As QUIC is now in its final stage of standardization, what to do with multi-path over QUIC has become one of the top questions discussed in the working group [14, 15, 51]. Even though

[19] See 5G signal coverage issues [36].

several proposals introduced multi-path capabilities in QUIC [11, 12, 52, 53], today, they did not address the concerns raised by the group regarding the deployability and benefits due to the lack of large-scale experimental study in real-world use cases [14]. We develop XLINK to answer these questions. Our study demonstrates the feasibility, deployability, and benefits of multi-path QUIC as an end-to-end service in commercial large-scale short video services.

Multi-path TCP. MPTCP was standardized in 2013[4], but to date, it is only available in a few mobile OSs [22]. The reason for the slowing adoption in the Internet is not only due to the fact that deploying MPTCP is hard in practice [6], but also the performance issues caused by problems such as MP-HoL blocking [6, 17] and concerns about the costs when aggregation is enabled [16]. A large body of work on MPTCP from in-lab controlled experiments revealed that MPTCP could outperform single-path TCP, but many factors (e.g., download size, the disparity between two paths) affected the performance [43, 54]. Indeed, we argue that multi-path features needed by different applications can vary significantly. Hence, a user-spaced multi-path approach that collaborates closely with an application is a pivotal step to move forward. XLINK, tailored for short videos, demonstrates the power of such an application-driven approach in multi-path.

Packet scheduling in multi-path. A multi-path packet scheduler is the most critical component that impacts any multi-path transport performance. The default MPTCP implementation selects the path with the lowest RTT from those with available congestion windows and uses opportunistic retransmission and penalization to mitigate the HoL-blocking [6]. Penalization lowers down the aggregated capacity. Several improvements have been proposed to address the limitations of penalization [18, 19, 55] based on network predictions. However, these estimations vary significantly in highly mobile scenarios and thus are not accurate enough. Both STMS [20] and DEMS [8] achieve in-order-receive through out-order-sending. STMS estimates the gap in packet sequence number between the paths, while DEMS decouples sub-flows by splitting a data chunk into two sub-flows. Other low latency MPTCP solutions (e.g., [56]) do not apply to video services as they achieve low latency through a large amount of redundancy. XLINK differs from these approaches. It is designed with scalability and deployability in mind, leverages real-time remote QoE-feedback to overcome HoL blockings, and balances performance and cost. Moreover, XLINK's scheduler is video-centric, supporting the expression of video QoE-awareness with priority-based re-injections at both stream and video-frame levels.

Cross-layer video improvement. XLINK is also closely related to cross-layer video QoE enhancement techniques [41, 57–60], such as DASH [57–59] and RTC [41, 60]. XLINK is inspired by these past works but differs from them in that instead of the bit-rate adaptation techniques that are limited to a single path's capacity, XLINK applies QoE control to multi-path adaptation, which cost-efficiently aggregates multiple paths' bandwidth. We also notice that bit-rate adaptation can be applied in multi-path. For example, MP-DASH [46] introduces MPTCP support for DASH, but as MPTCP lives in the kernel, a tight integration of DASH and MPTCP is not easy [47]. Thus MP-DASH uses a coarse-grained decision logic which decides whether the cellular sub-flow should be activated or not. Meanwhile, the multi-path scheduler in [61] is also coarse grained where it simply prioritizes video packets in MPTCP. In contrast, XLINK's control is much more fine grained in time (hundreds of milliseconds), and we believe the XLINK's feedback-based fine-grained adaptation could serve as a powerful, flexible and easy-to-use framework for future multi-path research.

9 DISCUSSIONS AND LIMITATIONS

Cellular cost: The client's cellular cost is another important factor that may affect the adoption of multi-path transport. As XLINK is integrated as a part of a phone app, we provide two solutions. (1) The first one is a zero-rating service in our app, for which we collaborate with mobile carriers so that customers enrolled in a special data plan can use our app at free charge. (2) The second one is a switch button in the app so a user can decide to turn on XLINK when needed.

Congestion control fairness: Our current implementation of XLINK uses "decoupled" Cubic congestion control, as in other mobile multi-path transport [46, 47]. The reason behind this is that WiFi and cellular networks are unlikely to share the bottleneck link that is often the "last mile" for wireless networks; this even holds for 5G NSA as reported in [36]. However, with the deployment of 5G SA, there is a possibility that the bottleneck moves from the "last mile" to other parts of the network (e.g., close to CDN servers) and two paths share the bottleneck. In this case, the coupled variant is preferred for fairness [62]. While the focus of this paper is not congestion control, the congestion control fairness of XLINK is worth further investigation.

10 CONCLUSION

Despite the vast interests from research, large-scale deployments of multi-path transport have been slow over the past decade in the public Internet. We believe that the emergence of QUIC as an end-to-end solution has brought the critical opportunity to change the landscape. We present XLINK, a QoE-driven multi-path transport implemented as a lightweight extension over QUIC, and conducted a large-scale experimental study in our e-commerce short video services. With XLINK, we proved the feasibility, deployability and benefits of multi-path QUIC. We believe that the implications of such a QoE-driven approach extend beyond short videos and pave the way for exciting new avenues for exploration in multi-path transport such as long-form VoDs, live-streaming, AR, and VR.

ACKNOWLEDGMENTS

We are grateful to the our shepherd Srinivasan Seshan and the anonymous reviewers for their insightful comments and feedbacks, which have greatly improved the quality of this paper.

We also thank Christian Huitema for his vital contribution on the multi-path protocol design.

We thank all teams at Alibaba that help deploy XLINK, especially Yongchao Lao and Yongguang Wang in the media platform team, Qing Yang and Liangwei Shen in the mobile platform team, and Hongyu Guo in the Alibaba mobile technology group.

Zhenyu Li's work was partially supported by Beijing Natural Science Foundation No. JQ20024 and Natural Science Foundation of China NO. U20A20180.

REFERENCES

[1] Amazon product videos. https://videoreviewlabs.com/amazon-product-videos/, 2020.

[2] Target China. WHAT IS "INTERNET CELEBRITY ECONOMY" IN CHINA. https://targetchina.com.au/article/internet-celebrity, 2020.

[3] Wikipedia. Ryan's World. https://en.wikipedia.org/wiki/Ryan%27s_World, 2020.

[4] Alan Ford, Costin Raiciu, Mark J. Handley, and Olivier Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, January 2013.

[5] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z Morley Mao. Mp-h2: a client-only multipath solution for http/2. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[6] 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 *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pages 399–412, 2012.

[7] Quentin De Coninck and Olivier Bonaventure. Multipath Extensions for QUIC (MP-QUIC). Internet-Draft draft-deconinck-quic-multipath-06, Internet Engineering Task Force, November 2020. Work in Progress.

[8] Yihua Ethan Guo, Ashkan Nikravesh, Z Morley Mao, Feng Qian, and Subhabrata Sen. Accelerating multipath transport through balanced subflow completion. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 141–153, 2017.

[9] List of Android smartphones. https://en.wikipedia.org/wiki/List_of_Android_smartphones, 2020.

[10] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 183–196, 2017.

[11] Yanmei Liu, Yunfei Ma, Christian Huitema, Qing An, and Zhenyu Li. Multipath Extension for QUIC. Internet-Draft draft-liu-multipath-quic-02, Internet Engineering Task Force, December 2020. Work in Progress.

[12] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *Proceedings of the 13th international conference on emerging networking experiments and technologies*, pages 160–166, 2017.

[13] Christian Huitema. QUIC Multipath Negotiation Option. Internet-Draft draft-huitema-quic-mpath-option-00, Internet Engineering Task Force, October 2020. Work in Progress.

[14] IETF mail archive. https://mailarchive.ietf.org/arch/browse/quic/, 2020.

[15] Spencer Dawkins. Questions for Multiple Paths In QUIC. Internet-Draft draft-dawkins-quic-multipath-questions-01, Internet Engineering Task Force, January 2021. Work in Progress.

[16] Improving Network Reliability Using Multipath TCP. https://developer.apple.com/documentation/foundation/urlsessionconfiguration/improving_network_reliability_using_multipath_tcp, 2020.

[17] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. A measurement study on multi-path tcp with multiple cellular carriers on high speed rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, 2018.

[18] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, 2017.

[19] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli. Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, 2016.

[20] Hang Shi, Yong Cui, Xin Wang, Yuming Hu, Minglong Dai, Fanzhao Wang, and Kai Zheng. STMS: Improving MPTCP throughput under heterogeneous networks. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018.

[21] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 459–471, 2013.

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

[23] ACM code of ethics and professional conduct. https://www.acm.org/code-of-ethics, 2020.

[24] A.O. Scott. Two minute geniuses. https://www.nytimes.com/interactive/2020/12/09/magazine/tiktok-twitter-internet-video.html, 2020.

[25] Promote your eBay items with videos. https://www.thebalancesmb.com/promote-your-items-with-ebay-videos-1140577, 2019.

[26] Redfin. https://www.redfin.com/, 2020.

[27] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, 2012.

[28] Recommended video bitrates for HDR uploads. https://support.google.com/youtube/answer/1722171?hl=en#zippy=%2Cbitrate, 2020.

[29] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. A first look at quic in the wild. In *International Conference on Passive and Active Network Measurement*, pages 255–268. Springer, 2018.

[30] How Facebook is bringing QUIC to billions. https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions, 2020.

[31] Multipath TCP and load balancers. http://blog.multipath-tcp.org/blog/html/2018/10/04/mptcp_load_balancers.html, 2018.

[32] Quentin De Coninck and Olivier Bonaventure. Observing network handovers with multipath tcp. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 54–56, 2018.

[33] Alireza Keshavarz-Haddad, Ehsan Aryafar, Michael Wang, and Mung Chiang. Hetnets selection by clients: convergence, efficiency, and practicality. *IEEE/ACM Transactions on Networking*, 25(1):406–419, 2016.

[34] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-Draft draft-ietf-quic-transport-33, Internet Engineering Task Force, December 2020. Work in Progress.

[35] Mansoor Shafi, Andreas F Molisch, Peter J Smith, Thomas Haustein, Peiying Zhu, Prasan De Silva, Fredrik Tufvesson, Anass Benjebbour, and Gerhard Wunder. 5g: A tutorial overview of standards, trials, challenges, deployment, and practice. *IEEE journal on selected areas in communications*, 35(6):1201–1221, 2017.

[36] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 479–494, 2020.

[37] 5G vs. Wi-Fi 6: A Powerful Combination for Wireless. https://www.intel.com/content/www/us/en/wireless-network/5g-technology/5g-vs-wifi.html, 2020.

[38] 3GPP release 16. https://www.3gpp.org/release-16, 2020.

[39] Mohamed Boucadair, Olivier Bonaventure, Maxime Piraux, Quentin De Coninck, Spencer Dawkins, Mirja Kühlewind, Markus Amend, Andreas Kassler, Qing An, Nicolas Keukeleire, and SungHoon Seo. 3GPP Access Traffic Steering Switching and Splitting (ATSSS) - Overview for IETF Participants. Internet-Draft draft-bonaventure-quic-atsss-overview-00, Internet Engineering Task Force, May 2020. Work in Progress.

[40] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate record-and-replay for {HTTP}. In *2015 {USENIX} Annual Technical Conference ({USENIX}{ATC} 15)*, pages 417–429, 2015.

[41] 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 *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 267–282, 2018.

[42] Amazon CloudFront pricing. https://aws.amazon.com/cloudfront/pricing/, 2020.

[43] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. Wifi, lte, or both? measuring multi-homed wireless internet performance. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 181–194, 2014.

[44] Martin Duke and Nick Banks. QUIC-LB: Generating Routable QUIC Connection IDs. Internet-Draft draft-duke-quic-load-balancers-06, Internet Engineering Task Force, November 2019. Work in Progress.

[45] XQUIC is the Alibaba's implementation of the IETF QUIC protocol. https://xquic.org/, 2020.

[46] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 129–143, 2016.

[47] Bo Han, Feng Qian, Shuai Hao, and Lusheng Ji. An anatomy of mobile web performance over multipath tcp. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, 2015.

[48] . https://developer.android.com/reference/android/os/BatteryManager, 2020.

[49] QUIC at 10,000 feet. https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/edit, 2020.

[50] QUIC IETF working group. https://datatracker.ietf.org/wg/quic/about/, 2020.

[51] Spencer Dawkins. What To Do With Multiple Active Paths in QUIC. Internet-Draft draft-dawkins-quic-what-to-do-with-multipath-03, Internet Engineering Task Force, January 2021. Work in Progress.

[52] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, and R. Steinmetz. Multipath quic: A deployable multipath transport protocol. In *2018 IEEE International Conference on Communications (ICC)*, 2018.

[53] Quentin De Coninck, François Michel, Maxime Piraux, Florentin Rochet, Thomas Given-Wilson, Axel Legay, Olivier Pereira, and Olivier Bonaventure. Pluginizing quic. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, 2019.

[54] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. A measurement-based study of multipath tcp performance over wireless networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, 2013.
[55] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. Musher: An agile multipath-tcp scheduler for dual-band 802.11ad/ac wireless lans. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, 2019.
[56] HyunJong Lee, Jason Flinn, and Basavaraj Tonshal. Raven: Improving interactive latency for the connected car. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, 2018.
[57] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, 2017.
[58] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. 2015.
[59] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, 2014.
[60] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.
[61] Xavier Corbillon, Ramon Aparicio-Pardo, Nicolas Kuhn, Géraldine Texier, and Gwendal Simon. Cross-layer scheduler for video streaming over mptcp. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, 2016.
[62] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, 2011.
[63] R. Netravali A. Sivaraman and K. J. Winstein. Mpshell. https://github.com/ravinet/mahimahi/releases/tag/old%2Fmpshell_scripted, 2020.
[64] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 459–471, 2013.

APPENDIX

Appendices are supporting material that has not been peer reviewed.

A CROSS ISP PATH DELAY INCREASE

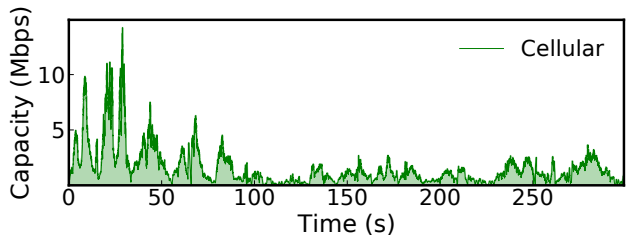
The table of cross ISP path delay increase to our CDN server is shown below. A, B and C denotes represent three anonymized ISPs.

Table 4: Relative increase of cross-ISP LTE delay (%)

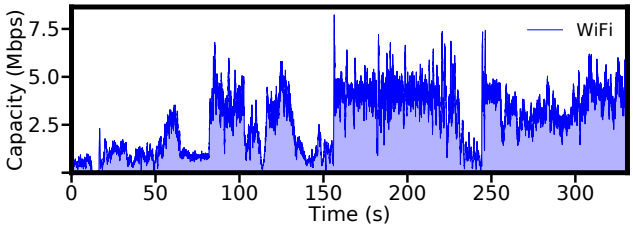
ISP	A	B	C
A	0	21%	17%
B	42%	0	54%
C	39%	34%	0

B METHODS ON TRACE-DRIVEN EVALUATION

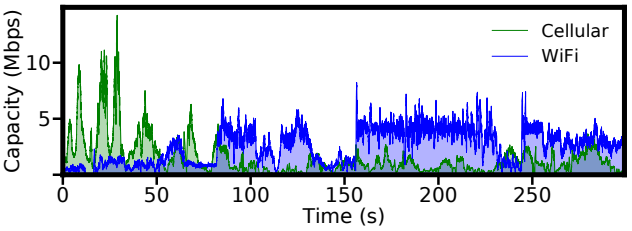
In addition to tests in real-world deployment, we used the multi-path extension of Mahimahi, *mpshell* [63], as our network emulator to understand the inner dynamics of multi-path transport. Thanks to its highly accurate network link emulation based on packet delivery traces collected in realistic networks, we were able to unveil the low-level details about why vanilla-MPQUIC failed to timely adapt to network changes. We could also quickly verify the results of XLINK's QoE-driven multi-path scheduling and management strategies, and compared and contrasted XLINK with other schemes in a reproducible way.



(a) Cellular trace collected on high speed rail.



(b) Onboard WiFi trace collected on high speed rail.



(c) Multi-path traces combined (a) and (b).

Figure 15: Trace examples: (a-b) Single path trace. (c) Multi-path trace.

Trace collection. We used *saturatr* [64] to collect network link traces in various environments. Figure 15a–15b are two examples of our traces that we collected for our extreme mobility evaluation. In our multi-path experiment, we always replayed different traces collected in the same environment on different paths. Figure 15c shows an example of a pair of multi-path traces.

Experiments with Mahimahi. In order to compare with other multi-path protocols with Mahimahi, we implemented a simple "video player" in C, which sequentially requested data chunks from a web-server and consumed received data at a constant bit-rate, which was configurable (to emulate playing videos with different resolutions). The video player ran inside *mpshell*, accessed our web-server via emulated links. Both the player and the web-server could be configured to run on top of different transport schemes (e.g., XLINK, vanilla-MPQUIC, and MPTCP). With this setup, we were able to evaluate XLINK and other schemes with consistent metrics.

C ACK_MP FRAME FORMAT

The structure of ACK_MP frame is shown in Fig. 16.

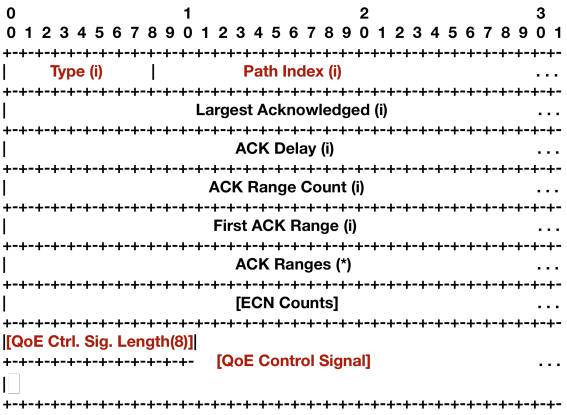


Figure 16: ACK_MP frame extension