AggDeliv: Aggregating Multiple Wireless Links for Efficient Mobile Live Video Delivery

Jinlong E[†‡*], Lin He[‡¶], Zongyi Zhao[‡], Yachen Wang[§], Gonglong Chen[§], Wei Chen[§]
[†] Renmin University of China [‡] Tsinghua University [§] Tencent [¶] Zhongguancun Laboratory, China
[*] Engineering Research Center of Database and Business Intelligence, MOE, China
ejinlong@ruc.edu.cn, he-lin@tsinghua.edu.cn, xinshengzzy@foxmail.com,
{yachenwang, garinchen, allenwchen}@tencent.com

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621184

***Abstract*—Mobile live-streaming applications with stringent latency and bandwidth requirements have gained tremendous attention in recent years. Encountered with bandwidth insufficiency and congestion instability of the wireless uplinks, multi-access networking provides opportunities to achieve fast and robust connectivity. However, the state-of-the-art multi-path transmission solutions are lack of adaptivity to the heterogeneous and dynamic nature of wireless networks. Meanwhile, the indispensable video coding and transformation bring about extra latency and make the video delivery vulnerable to network throughput fluctuation. This paper presents AggDeliv, a framework that provides efficient and robust multi-path transmission for mobile live video delivery. The key idea is to relate multi-path packet scheduling to congestion control optimization over diverse wireless links and adapt it to the mobile video characteristics. This is achieved by probabilistic packet allocation based on links' congestion windows, wireless-oriented delay and loss aware congestion control, as well as lightweight video frame coding and network-adaptive frame-packet transformation. Real-world evaluations demonstrate that our framework significantly outperforms the state-of-the-art solutions on aggregate goodput and streaming video bitrate.**

I. Introduction

Recent years have witnessed the popularity of a variety of mobile live-streaming applications, especially interactive live streaming on e-commerce, sports, and education, which generally impose stringent latency requirements (10 ms$\sim$100 ms). In addition, consumers' appetites for video resolution are moving towards 4K and beyond, termed as *ultra high-definition* (UHD). To enable low-latency and high-bandwidth mobile live video delivery, camera technology, computing power, and wireless resources are all key factors. While the former two factors have seen significant improvements, wireless uplinks often have limited bandwidth (*e.g.,* only 50 $\sim$100 Mbps) [1] and prevalently experience communication failures caused by mobile operators' malfunctions [2]. The efficiency goal of live video delivery urges mobile ingest clients to overcome bandwidth deficiencies and unstable connections.

Fortunately, today's mobile devices can simultaneously access the Internet through multiple types of networks (*e.g.,* WiFi, LTE/5G, LoRa), which provides opportunities to achieve fast and robust connectivity. In reality, multi-path transport has gained tremendous attention over the last decade. The most widely-used protocol MPTCP [3], however, requires OS-level support in smartphones, whose high cost severely impedes the deployment of mobile apps. Taking advantage of QUIC's properties that can improve transmission efficiency, multi-path extensions for QUIC (termed MPQUIC) have become a focus of attention in both academia and industry [4]–[6].

A fundamental factor affecting MPQUIC's transmission performance is packet scheduling, *i.e.,* distributing network traffic across multiple available paths. The default scheme learned from MPTCP is to prefer the path with the lowest round-trip time (RTT) for packet sending. However, it overlooks the widespread path heterogeneity in wireless networks as well as frequent handoffs caused by user mobility, which probably leads to multi-path *head-of-line blocking* [7] in mobile environments and incurs serious reordering at the receiver side. A seemingly feasible approach to alleviating the hurdle is to opportunistically reinject the blocked packets to other paths [7], [8]. Still, the increase in transmission traffic brings unneglectable cost overhead to the network, especially for large-scale services. Some studies on MPTCP propose more sophisticated scheduling schemes [9]–[11], whereas they rely too much on specific network characteristics (*e.g.,* bandwidth, RTT, loss) that are difficult to accurately estimate for wireless links with signal fluctuation.

Congestion control also plays a key role in the multi-path transport, which prevents network congestion by controlling the packet sending rate. Unfortunately, the existing congestion control algorithms may suffer from issues such as bufferbloat [12], [13] (*i.e.,* large receive queue) and starvation of flows [14] (*i.e.,* idle fast path) in dynamic environments due to mismatching the path diversities. Especially, the algorithms LIA [15] and OLIA [16] recommended by the up-to-date MPQUIC draft [6] employ fixed congestion window adjustment rules, which inadequately reflect RTT differences across multiple paths and are largely impacted by burst errors in wireless links.

In a nutshell, the state-of-the-art multi-path transmission solutions are lack of adaptivity to the heterogeneous and dynamic nature of wireless networks. Worse still, unlike common applications that involve only packet transmission, live video delivery requires additional frame coding. With the existing codecs such as H.264 [17] and HEVC [18], it generally takes a significant amount of time to encode and decode frames of a 4K and beyond video [1], [19]. Inter- and intra-frame data dependencies brought by encoding also make the video

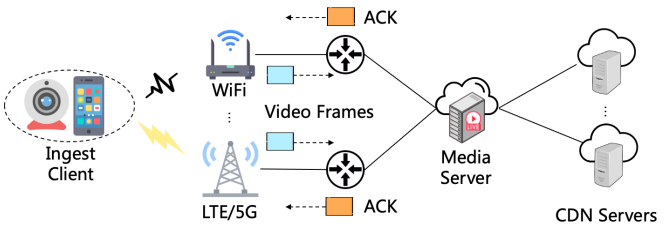

Fig. 1. Typical mobile live video delivery workflow with multi-path ingestion.

delivery vulnerable to network throughput fluctuations.

In this paper, we design a framework named AggDeliv on the basis of MPQUIC to achieve efficient and robust mobile live video delivery by aggregating multiple wireless links (§III-A). The key idea is to relate multi-path packet scheduling to congestion control optimization and adapt it to the mobile video characteristics. As depicted in Fig. 1, video frames captured at an ingest client (*e.g.,* a smartphone or an IP camera) can be streamed to the media server through multiple networks (typically WiFi and LTE/5G), which are then distributed to worldwide users by content delivery networks (CDNs). While the idea is promising and straightforward, we need to focus on the following three technical challenges to make it practical.

- *To properly grasp the heterogeneity of wireless links in packet scheduling*, we explore the present size proportions of the links' varying congestion windows. Accordingly, we design a *probabilistic* per-packet allocation mechanism following closely the variation of each link's transmission workload, which can conduct real-time packet processing while guaranteeing links' proportional workloads (§III-B).
- *To effectively adjust the congestion windows in line with wireless dynamics*, we ameliorate the mainstream approach to jointly consider in-flight packets and wireless errors of multiple links. Specifically, we devise a *delay-loss-aware* congestion control algorithm that takes into account the latest measured RTT and congestion level every time increasing and decreasing the congestion windows (§III-C).
- *To further achieve the timeliness of video frame transmission*, we match up the frame coding and frame-packet transformation speed with the network condition variations. Concretely, we conduct *pipelined* frame coding based on intra-frame layered averaging and differencing, and wisely retransmit or drop packets according to their playout deadline and the latest collected network characteristics (§III-D).

With the above enabling solutions, we implement AggDeliv to achieve high and stable throughput as well as low latency in the mobile environment. We conduct comprehensive experiments to evaluate AggDeliv both in a testbed with variable network settings and in the wild (§IV). In contrast to the state-of-the-art solutions, AggDeliv can improve the aggregate goodput up to 3.2×, and meanwhile reduce the packet transmission time by at least ∼11%, which is beneficial to data transmission of diverse mobile applications. When applying AggDeliv to live video delivery, it can yield an overall 22.3% higher bitrate than alternative approaches.

II. Background and Motivation

This section introduces the background of multi-path data transmission used for video delivery, followed by real-world performance measurements of the state-of-the-art solutions that motivate our study.

A. Multi-Path Transmission for Video Delivery

At present, various mobile applications have been cloudified on a large scale, which is able to be beneficial from multi-access data transmission. A typical example is the mobile live-streaming application, of which the multi-path video delivery workflow is depicted in Fig. 1. In view of the unaffordable traffic load of UHD videos for a single wireless uplink's bandwidth (∼100 Mbps for 5G and ∼50 Mbps for WiFi/LTE), the ingest client delivers video frames (encoded by H.264/HEVC) to the media server through multiple interfaces (WiFi module and cellular SIMs). Despite the promising prospect of multiple access, it is highly necessary to enable efficient and reliable switching of data transmission over multiple heterogeneous and dynamic wireless links.

As an alternative protocol to TCP released in recent years, QUIC improves transmission efficiency with optimization mechanisms such as simplified connection establishment and multi-level flow control [20]. In addition, as a user-space protocol, QUIC is apt to be deployed as part of various applications that are constantly evolving. In light of its virtues, extending QUIC to multi-path transport (*i.e.,* MPQUIC) has been in progress [6], in replace of the traditional MPTCP. This is helpful for transmitting large data from multi-homed devices like smartphones. Also, it provides resilience to connectivity failures for unstable wireless networks, given that the receiver can reorder and acknowledge the received packets from different paths.

The packet scheduler is a key module of MPQUIC in charge of choosing a proper path for the transmission of each packet. Instead of putting forward sole scheduling schemes, existing MPQUIC studies [4]–[6], [8] recommend reusing those of MPTCP, and refer to 4 commonly-used schemes:

- *Min-RTT* selects the path with the lowest RTT and available congestion window space to send packets.
- *Round Robin* simply sends each packet to a path in a round-robin manner that balances data to all paths.
- *BLEST* [9] waits for a fast path to transmit packets, so as to free up space in the sending window of a slow path.
- *ECF* [10] decides a path by estimating if the fast path has space to transmit more packets during the slow path's RTT.

These schemes' practical performance in heterogeneous and dynamic wireless networks remains to be examined.

Besides, congestion control should also be considered since it regulates QUIC packet transmission. Packets that increase the number of bytes in flight can only be sent when allowed by a path's congestion window $cwnd$. Most traditional TCP congestion control algorithms (*e.g.,* Reno [21], Veno [22], and CUBIC [23]) adjust the congestion window based on AIMD,

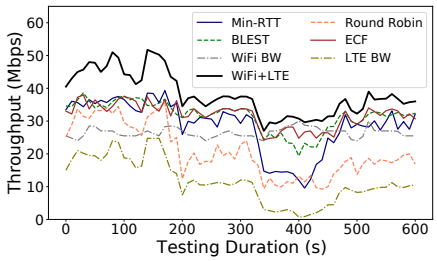

Fig. 2. Aggregate goodput variations of the state-of-the-art scheduling schemes.

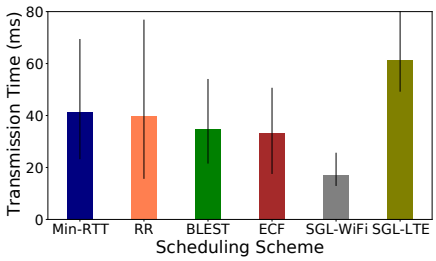

Fig. 3. Packet transmission time comparison among the state-of-the-art scheduling schemes.

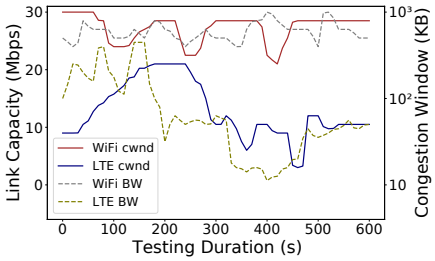

Fig. 4. Congestion window variations of two links during the testing period.

i.e., additively increasing $cwnd$ when an ACK is received and multiplicatively decreasing $cwnd$ when a packet loss occurs. Unlike these loss-based algorithms, the recently popular algorithm BBR [24] conducts congestion control according to the measured link capacity, which is shown to perform well in cellular networks [1]. However, all the algorithms separately maintain each subflow's congestion window when adopted for MPTCP, which makes the MPTCP flow unfairly take up more capacity than a single path TCP flow [3]. In contrast, the coupled congestion control algorithms LIA [15] and OLIA [16] tweak the AIMD principle for multi-path transport. They can avoid an unfair capacity distribution by the above uncoupled algorithms and are recommended by the latest MPQUIC draft [6]. As an evolution of LIA, OLIA has been demonstrated to better balance flows across the congested paths, but whether it fits the heterogeneous and fast changing wireless link conditions should be further tested.

B. Measurements and Observations

We next conduct measurements on the state-of-the-art packet scheduling and congestion control solutions described in §II-A, carefully examining if their real-world performance can boost practical data transmission across heterogeneous mobile networks. To ensure a consistent experimental environment, we replay the same pair of WiFi and LTE traces for data transmission with all solutions. Note that since many mobile ingest devices (*e.g.,* wireless IP cameras, driving recorders) have not been equipped with 5G, we are more concerned with LTE transmission, which may experience worse network conditions to be tackled. We collect the traces by walking on campus for a week, recording both networks' characteristics including available bandwidth, delay, and packet loss every 5 seconds. On this basis, we extract records in the peak hour for Internet access (around 3 p.m.), when the wireless links are prone to experience quality fluctuation.

We modify an open-source project [25] corresponding to the MPQUIC draft [6] by supplementing implementation of the schemes and algorithms described in §II-A. Then we conduct multi-path data transmission between two virtual machines (VMs) in the same cloud region with 4-core CPU@2.5 GHz and 16 GB of memory. Facilitated by a tool *iPerf* [26], the sender simultaneously makes full use of its two network interfaces to stream data to the receiver. We adopt another tool *tc* [27] to throttle the bandwidth and emulate the transmission delay fluctuation and packet loss rate on each network interface based on the WiFi and LTE trace records, respectively.

Measurement Results. We first conduct experiments on the state-of-the-art scheduling schemes Min-RTT, Round Robin (RR), BLEST, and ECF. To grasp the performance stability, we run each scheduling scheme with the extracted traces every weekday. The MPQUIC-recommended congestion control algorithm OLIA [16] serves for all the test items. We mainly focus on the following two metrics in our measurements:

- *Aggregate goodput* measures bits transmitted successfully through all network links per second. The practical value compared to aggregation of all links' available bandwidths can reflect the effectiveness of a packet scheduling scheme.
- *Packet transmission time* measures the time needed to transmit a packet between two devices, which indicates whether the sending path is properly selected. We average it over all the transmitted packets to observe the overall performance.

We monitor the aggregate goodput variations during testing with each scheduling scheme. Fig. 2 visualizes the results of all test items for a 10-minute period that achieves the overall best performance, together with the available bandwidths of WiFi and LTE links and their aggregation. Here we omit to show the variations in other periods which follow the similar fluctuating trend. We observe that the WiFi trace is relatively stable while the LTE trace changes rapidly, and such path heterogeneity has a significant influence on the performance of scheduling schemes. When the LTE's network condition is close to that of WiFi, all four schemes possess almost the same performance. Their aggregate goodputs all drop down as the network condition gap increases, among which the reduction magnitude of ECF is smaller than others but its gap with the aggregation of available bandwidths is visible.

We also measure packet transmission time over the above-configured wireless links. The average, maximum, and minimum results of the four scheduling schemes during all extracted periods are depicted in Fig. 3. Besides, we also give the counterparts when transmitting data with a single WiFi or LTE link for comparison. Obviously, the time for all scheduling schemes is not ideally short given the result of single-WiFi transmission. The maximum time values of Min-RTT and Round Robin are especially longer than those of the other schemes, as the two schemes are prone to cause head-of-line

TABLE I
PER-FRAME CODING TIME (MS) WITH MAINSTREAM CODECS.

Codec	Video Type	Encoding	Decoding	Total Time
H.264	HR	56.62	19.64	76.26
	LR	53.74	16.25	69.99
HEVC	HR	50.62	16.98	67.60
	LR	47.51	13.77	61.28

blocking when encountered with path heterogeneity. Although ECF and BLEST can help alleviate packet blocking through delayed processing by a fast path, they are very sensitive to the dynamics of wireless links, which leads to frequent improper packet allocation to a path with increasing RTT.

We further study the influence of congestion window adjustment when adopting MPQUIC for data transmission. Fig. 4 plots the $cwnd$ variation trends of both WiFi and LTE links with the default MPQUIC solution "Min-RTT + OLIA" [4] during the above throughput testing period, along with available bandwidths of the two links. While each link's congestion window variation trend is generally in line with that of its bandwidth, the $cwnd$ adjustment cannot closely follow the rapid change of throughput, especially the dramatic drop in the short period from 16 s to 20 s. As a result, the sender keeps transmitting packets to the link, which leads to severe head-of-line blocking that postpones the whole packet transmission.

To support live video delivery, we are also concerned with the performance of current mainstream video codecs H.264/AVC [17] and H.265/HEVC [18]. We conduct encoding and decoding on two uncoded 4K videos at 30 FPS downloaded from Xiph [28], each with the two codecs (implemented by FFmpeg [29]) in an Android smartphone with 8-core CPU@3.0 GHz and 8 GB of memory. The two videos are separately classified as high richness (HR) and low richness (LR) videos according to the judgment method in [19]. Table I shows the average encoding and decoding time of a single frame for the videos. We observe that the overall coding time (excluding transmission time) is beyond the frame playout deadline of 60 ms (as suggested in recent studies [19], [30]), which means that transmission of 4K videos requires extra optimization mechanisms.

Opportunity. The above measurements indicate that the state-of-the-art multi-path transmission solutions are inadequate to aggregate the available bandwidths of wireless links, and the root cause is their inapplicability to the networks' heterogeneity and dynamics. By observing that a congestion window's variation trend reflects the corresponding link's network condition in most time, we notice an opportunity to potentially address the unravelled issue – scheduling packets based on the paths' varying congestion windows and meanwhile optimizing MPQUIC's congestion control. On this basis, video characteristics like frame coding should further be elaborated to match the status of multi-path mobile networks.

III. AGGDELIV DESIGN

Guided by our observations, we design AggDeliv, a framework that provides efficient and robust multi-path data transmission for mobile live video delivery. We next describe our solution to achieve this goal.

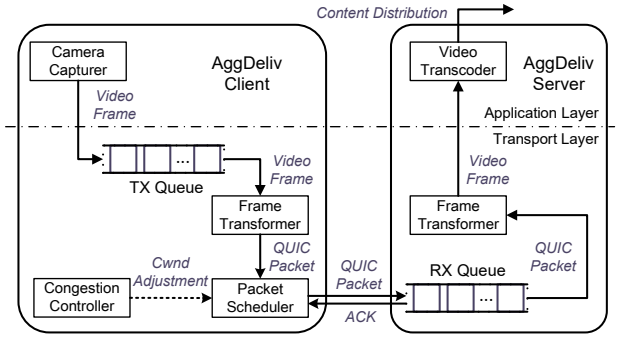

Fig. 5. An architectural overview of AggDeliv.

A. System Overview

Fig. 5 depicts the high-level architecture of AggDeliv. As a lightweight end-to-end framework, AggDeliv enables a mobile ingest client to communicate with a media server simultaneously through multiple wireless interfaces (*e.g.,* WiFi and cellular like LTE/5G). Taking the opportunity of QUIC as a user-space protocol, AggDeliv follows the cross-layer network designs [31] to closely integrate transport with the upper applications. The functionality and interaction of components on AggDeliv client and server are outlined as follows.

As a core component, *packet scheduler* is in charge of selecting a wireless link for the transmission of each ready packet, which wisely utilizes the size proportion of multi-path congestion windows for probabilistic packet allocation (§III-B). During the QUIC packet transmission over multiple links, *congestion controller* jointly adapts the congestion windows of all links based on their latest smoothed RTTs (SRTT) and congestion levels (§III-C). On this basis, the in-situ generated video from *camera capturer* is fast encoded to frames and cached in a TX queue on the client. *Frame transformer* fetches the frames from the queue and encapsulates them into QUIC packets in sequence. Meanwhile, its server counterpart reorders the QUIC packets received from multiple network links (cached in an RX queue) according to their sequence numbers and then repacks them into video frames (§III-D). Finally, *video transcoder* in the application layer transcodes the frames based on multiple resolutions and distributes them to CDN servers for nearby access.

B. Cwnd-Based Packet Scheduling

We start with designing a multi-path packet scheduling scheme to closely follow variations of all links' network conditions, instead of resorting to the on-the-fly estimation of network characteristics adopted by the state-of-the-art scheduling schemes.

Network Inference with Congestion Windows. To address the performance degradation problem caused by path heterogeneity, the key point is to grasp each link's network condition in real time and jointly consider all links for packet scheduling.

In light of congestion control algorithms [16], [24] probing network characteristics like RTT, bandwidth, and packet loss rate for the congestion window adjustment, we leverage a $cwnd$-based approach for network inference. To relieve the impact of burst errors in wireless networks, we do not directly take the instantaneous congestion window values as the network indicator. Instead, we propose the following approach to calculate each link's *periodical* congestion window.

Suppose there are totally m wireless links available for packet allocation, and their initial congestion windows are $\{cwnd_j^0\}, j = 1, 2, \cdots, m$. Whenever there occurs congestion window adjustment in a link, we record the updated $cwnd_j^i$ value as well as the adjustment time t_j^i. For a packet to be allocated at time t_c, we denote the congestion window affecting period as T_a. We calculate the periodical congestion window $\widetilde{cwnd}_j$ for each link j based on the congestion window values recorded during the affecting period, *i.e.,* $\{cwnd_j^k\}, t_j^k \in [t_c - T_a, t_c]$. Inspired by the real-time bandwidth estimation mechanism [32], we calculate the harmonic means of these recorded congestion windows. The difference is we further take the time from each $cwnd$ adjustment to the present as the corresponding $cwnd$ value's weight for the calculation, given that these $cwnd$ values are not recorded at equal intervals. Therefore, the periodical congestion window for the link j can be expressed by,

$$\widetilde{cwnd}_j = n_a / \sum_k (t_c - t_j^k)/cwnd_j^k, \tag{1}$$

where n_a is the number of times the link's congestion window is adjusted in the affecting period.

Probabilistic Packet Allocation. Through the above efforts, we can continuously infer each wireless link's network condition with its congestion windows. On this basis, we consider packet allocation to all available links according to the size proportion of their periodical congestion windows $\{\widetilde{cwnd}_j\}(j = 1, 2, \cdots, m)$. In practice, to improve the timeliness of packet processing, we design a *probabilistic* mechanism to enable per-packet allocation while guaranteeing the links' proportional transmission workloads.

We first define *routing probability* p_j^r as the probability that a packet is allocated to a certain wireless link j. We can calculate a distribution of routing probabilities $\{p_j^r\}(j = 1, 2, \cdots, m)$ based on the proportions of their respective congestion windows in the total congestion window size,

$$p_j^r = \widetilde{cwnd}_j / \sum_{u=1}^{m} \widetilde{cwnd}_u, \ \sum_{j=1}^{m} p_j^r = 1. \tag{2}$$

Then we allocate packets one after another to the wireless links in a *weighted round robin* manner, in which the routing probabilities are deemed as path weights. A succession of probability intervals is formed by successive accumulation of each path weight, which can be represented by $[0, p_1^r), [p_1^r, p_1^r + p_2^r), \cdots, [\sum_{j=1}^{m-1} p_j^r, 1]$ $(\sum_{j=1}^{m} p_j^r = 1)$. For each packet, we randomly generate a value x in the range $(0, 1)$ and observe which probability interval it lies in. If $x \in [\sum_{j=1}^{v-1} p_j^r, \sum_{j=1}^{v} p_j^r)$, the link v will be selected to transmit the packet. By this means, the overall traffic distribution is corresponding to the routing probabilities.

C. Delay-Loss-Aware Congestion Control

The above-described packet scheduling scheme attempts to capture the changes in network conditions by online calculating the links' periodical congestion windows, which helps address the performance degradation resulting from path heterogeneity. However, MPQUIC's congestion control algorithms, represented by OLIA, are inadaptable to wireless link dynamics. We next design an algorithm that takes both delay and loss factors into account when adjusting the congestion windows.

RTT-Based Congestion Window Adjustment. As a congestion control algorithm recommended by MPQUIC, OLIA provides good performance in the wired network environment [4], which determines a link's network condition mainly based on its throughput variation. Unfortunately, the algorithm fails to achieve the desired performance in the heterogeneous and dynamic wireless networks (as shown in §II-B) due to inadequate consideration of RTT diversity across multiple links that leads to packet disorder. To reduce the delay difference, we ameliorate OLIA by introducing RTT as a determinant of the congestion window adjustment.

Concretely, for a link with extremely high RTT, we decrease its congestion window based on an exponentially weighted moving average, which can reduce the impact of fast $cwnd$ decrease to the multi-path aggregate goodput. We define an RTT limiting factor γ as the ratio of a link's RTT to the latest minimum RTT measured across all links. If a link j's limiting factor γ_j $(= RTT_j/RTT_{min})$ exceeds a threshold θ_γ, we update its congestion window to

$$cwnd_j^{t+1} = \beta * cwnd_j^t + (1 - \beta) * (cwnd_j^t/\gamma_j), \tag{3}$$

where $cwnd_j^t$ and $cwnd_j^{t+1}$ are the link's old and new congestion windows respectively, and β is a decay factor. Especially, when $cwnd_j^t$ is still in the slow start phase, we further reset the slow-start threshold to $cwnd_j^t$. When $\gamma_j < \theta_\gamma$ (*i.e.,* the link j's RTT is not high), we follow the mechanism of congestion window increase adopted by OLIA,

$$cwnd_j^{t+1} = cwnd_j^t + \frac{(cwnd_j^t/{RTT_j}^2)}{(\sum_j cwnd_j^t/RTT_j)^2} + \alpha/cwnd_j^t, \tag{4}$$

where α represents the opportunistic linked increase weight of link j relative to the optimal link (specified in [16]).

Loss-Based Congestion Level Inference. Given that mainstream congestion control algorithms like OLIA are oftentimes impacted by burst errors of wireless networks, it is a high priority to accurately discriminate wireless random packet loss from congestion loss. Inspired by TCP Veno [22], we judge the network condition by the number of packets retained in the routing queue. Differently, we also consider congestion caused

by wireless burst traffic and calculate the number of packets cached in each link's queue by

$$N_{cg} = (cwnd/RTT - cwnd/RTT_{cg}) * RTT_{min}, \quad (5)$$

where RTT_{cg} can be gradually collected from the smoothed RTTs (*i.e.,* $SRTT$s) when congestion losses occur by

$$RTT_{cg} = (1-\beta) * RTT_{cg} + \beta * SRTT. \quad (6)$$

If N_{cg} is larger than a congestion threshold θ_c, the present loss is deemed as a random packet loss; otherwise, it is a congestion loss. We thus update the link j's congestion window with different decrease weights w_d^r and w_d^c ($w_d^r < w_d^c$) in the two cases by

$$cwnd_j^{t+1} = \begin{cases} (1-w_d^r) * cwnd_j^t, & N_{cg} > \theta_c, \\ (1-w_d^c) * cwnd_j^t, & N_{cg} \leq \theta_c. \end{cases} \quad (7)$$

To help achieve a high aggregate goodput, we also maintain large congestion windows for each link by drawing lessons from TCP Veno's congestion avoidance phase. The congestion level is taken to be high when $N_{cg} \leq \theta_c$, and accordingly, we keep not increasing the congestion window until the sender receives two normal ACKs. Otherwise, we increase the congestion window once an ACK is received as usual.

D. Network-Adaptive Video Transmission

Even though our packet scheduling scheme together with congestion control amelioration boosts the performance of multi-path data transmission, further optimization is necessary for live video delivery to address its time-consuming frame coding and frame transmission's vulnerability to network throughput fluctuations. To this end, we design cost-efficient frame coding and frame-packet transformation mechanisms.

Pipelined Frame Coding. Given that the overall frame coding time of a 4K and beyond video with existing codecs (*e.g.,* H.264 [17], HEVC [18]) is beyond the suggested frame playout deadline (as shown in §II-B), we devise a fast and lightweight coding approach for video ingestion. An original frame captured from the camera is generally in the YUV420 format, where four pixels are represented with 4 luminance (Y) values and 2 chrominance (UV) values. Instead of simultaneously conducting inter- and intra-frame coding as in the mainstream codecs, we iteratively divide every single frame into hierarchical blocks and calculate pixel averages and their differences to relieve the transmission time and traffic.

Specifically, a frame is initially divided into non-overlapping blocks of $n \times n$ pixels ($n = 4, 8$ for 4K and 8K videos, respectively), and the average values of these blocks form a 540P video, which are denoted as $\{A_{ij}\}$ ($i = 1, 2, \cdots, 540, j = 1, 2, \cdots, 960$). We further iteratively divide each block into four blocks until the newly generated blocks only have one pixel, and then compute the averages of all pixels in each divided block $\{A_{ijk}\}, \{A_{ijkl}\}, \cdots$ ($k, l = 1, 2, 3, 4$). On this basis, we calculate the differences between the averages in two adjacent layers, *e.g.,* $\Delta_{ijk} = A_{ijk} - A_{ij}$. Note that only the first three differences in each layer are calculated, since the fourth difference can be deduced by them together with the top-layer average value. By this means, the ingest client need only send the top-layer averages and all calculated differences in succession, by which the media server can reconstruct each pixel of the original frame (*i.e.,* the above averages in the lowest layer). For the pixel (i, j, k, l) of a 4K video, this process can be represented as

$$A_{ijkl} = A_{ij} + \Delta_{ijk} + \Delta_{ijkl}. \quad (8)$$

We can also reconstruct an 8K video by calculating $\{A_{ijklm}\}$ ($m = 1, 2, 3, 4$) with Eq. (8) adding a sum term Δ_{ijklm}.

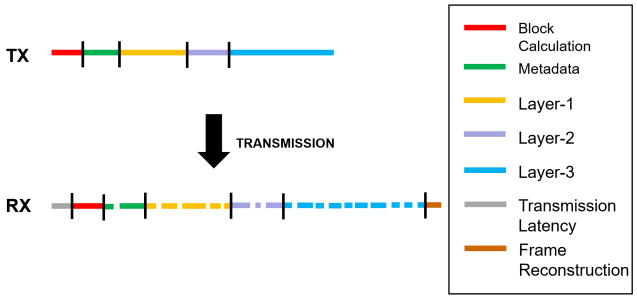

Fig. 6. A pipelining scheme for video frame coding.

In addition, the existing codecs commonly execute frame-level coding and transmission in a *stop-and-wait* manner. The sender starts transmission only after the whole frame has been encoded, and the receiver conducts frame decoding after receiving all involved parts. Obviously, it is an inefficient video delivery approach that incurs extra delays. In contrast, our mechanism enables frame-level encoding, transmission, and decoding in a *pipelining* manner owing to independent relationships across successive frames and only forward dependency among multi-layer differences in each frame. As shown in Fig. 6, the sender can start transmission as soon as the top-layer average and difference calculation is finished, and then transmits differences in each layer once it is encoded. The differences in a layer are ready to be decoded as soon as they arrive at the receiver, since the dependent data that are sent earlier should be already available as expected. Accordingly, the receiver pipelines the decoding process with the data receiving process and reconstructs a frame when all parts have been received and decoded.

Adaptive Frame-Packet Transformation. To leverage MPQUIC for multi-path data transmission, the video frames need to be encapsulated into QUIC packets on the sender and transformed back on the receiver. We expect the size of each QUIC packet to be the protocol's maximum segment size (MSS), which fully utilizes the payloads and facilitates packet scheduling. After a frame is encoded into a series of averages and differences, we continuously extract the maximum payload for a packet from the frame parts in sequence. Note that each frame parts (*i.e.,* averages or differences in a layer) can generally be divided into a number of payloads, and a packet can be encapsulated with payload across frame parts (the last packet for a frame may have insufficient payload). Given that

a frame part should be decoded as a whole that is influenced by any packet loss, we particularly schedule them together to an available wireless link, *i.e.,* the packet from different frame parts are transmitted along with the upper-layer packets.

To support live video delivery, the deadline for transmission to the media server should be less than half the time of the frame playout deadline (60 ms suggested by the recent studies [19], [30]). The header information of a packet indicates which frame it belongs to and the corresponding deadline. To minimize the link's bandwidth wastage, packets received past their deadlines should be dropped. Before transmitting each packet p_i, the sender estimates whether it could be delivered within its deadline T_d according to its size s_i and the selected link's throughput r_t and delay (or roughly half RTT) d_t based on the latest transmission. Suppose the present time is t_c and the generation time of packet p_i is t_g^i, then we drop the packet as well as the remaining packets belonging to the same frame when $t_c - t_g^i + s_i/r_t + d_t > T_d$.

Besides, when a wireless link transmitting the top-layer packets fails or becomes congested, the whole frame's decoding and reconstruction will be postponed. We address this issue by reallocating the top-layer packets to another available link based on our $cwnd$-based packet scheduling scheme. Note that we conduct rescheduling only if no packet is transmitted successfully on a link for a duration $T_\theta = 1/5(T_d - t_c + t_g^i)$ and the packets are still within their deadline. To ensure that high-layer packets are received prior to their sub-layer packets, the latter's transmission will be preempted by that of the former when they are allocated to the same link.

IV. Evaluation

In this section, we first briefly present the implementation of AggDeliv, and then demonstrate its practical data transmission efficiency with both data trace driven evaluations and live video delivery experiments in the wild.

A. System Implementation

According to the design architecture in Fig. 5, we implement all AggDeliv components across the ingest client and media server in Golang. For multi-path transmission, we reuse most of the open-source MPQUIC project [25] mentioned in §II-B. Particularly, we implement our cwnd-based scheduling scheme in replace of the default scheme Min-RTT, and also ameliorate the congestion control algorithm OLIA by adding delay and loss aware mechanisms. As suggested by [23], [30], we set the decay factor β in Eq. (3) and the decrease weights w_d^r and w_d^c in Eq. (7) to be 0.2, 0.2, and 0.3, respectively. We further select optimal values for the determinative parameters, including congestion window affecting period T_a, limiting factor threshold θ_γ, and congestion threshold θ_c, by grid search (as shown in §IV-C).

On this basis, AggDeliv client is integrated with a video capture application through stream transmission, which encodes and transforms the live captured video based on our designed mechanisms. The frame delivery deadline is set to be 30 ms, half of the suggested frame playout deadline (*cf.* §III-D). Note that we retain the original PEM coding for audio frames, since they are much smaller in size and inadaptable to our image-specific coding. As a counterpart, AggDeliv server reversely processes the delivered video (*i.e.,* transforming back and decoding), and it supports re-encoding the video with mainstream codecs like H.264 or HEVC (and audio codecs like AAC [33]). The modules in both platforms are implemented as separate processes, which interact with each other by a popular procedure call framework gRPC [34].

B. Experiment Setup

To compare AggDeliv with the state-of-the-art multi-path transmission solutions, we deploy AggDeliv client and server on the two cloud VMs used for measurements in §II-B. Likewise, we conduct the network trace replay experiments with AggDeliv, in which data is streamed across two paths with network conditions set according to our collected traces (by tools *iPerf* [26] and *tc* [27]). Given the AggDeliv's goal to achieve efficient multi-path transmission, we still regard the two metrics *aggregate goodput* and *packet transmission time* during its measurements. Among the previous measurement results, ECF [10] achieves the best performance among the state-of-the-art multi-path scheduling schemes. We take it together with Min-RTT [4], the basic scheduling scheme in both MPTCP and MPQUIC, as baselines for comparison, and ignore another two schemes, Round Robin [5] and BLEST [9], with moderate performance. As for comparative congestion control algorithms, we adopt the MPQUIC-recommended OLIA [16] and the popular BBR [24]. By combining these scheduling schemes and congestion control algorithms together with our solution, we finally formulate 5 test items.

To evaluate AggDeliv's real-world performance for mobile live video delivery, we take the aforementioned cloud VM with a public IP address as AggDeliv server and deploy AggDeliv client on an Android smartphone (with the configuration identical to that in §II-B). The two devices serve as the ingest client and media server for video delivery, respectively, which communicates through both WiFi and cellular networks in the wild. As illustrated in §II-B, we adopt LTE instead of 5G for testing to cover the network environments of low-end mobile ingest devices. We download 5 uncoded 4K videos at 30 FPS in the format of YUV420 from a collection of video test media under Xiph [28]. According to the judgment method in [19], two videos are classified as high richness (HR) videos and the rest three are low richness (LR) videos. We use these videos in turn as live source videos, and unify their playback duration to 10 minutes by concatenating each video to itself. Each video is delivered in two mobility patterns – the ingest client keeps static in a place or is taken by a person who walks around the campus. We adopt MP-DASH [35] and XLINK [8] that applies MPTCP and MPQUIC to video streaming, respectively, as alternative approaches for the delivery testing.

C. Performance on Multi-Path Transmission

We start with measuring AggDeliv's performance on generic data transmission by metrics aggregate goodput and packet

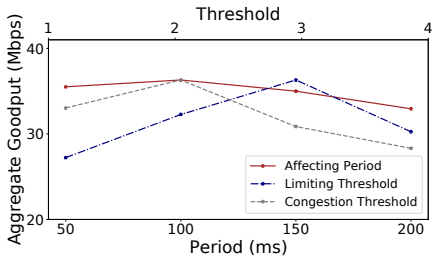

Fig. 7. Parameter selections to achieve the optimal aggregate goodput.

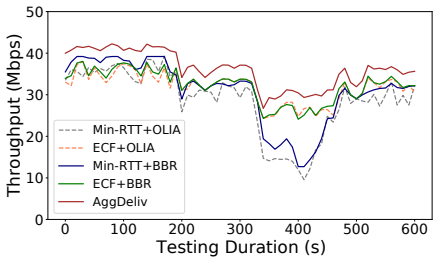

Fig. 8. Aggregate goodput variations of AggDeliv and the comparative approaches.

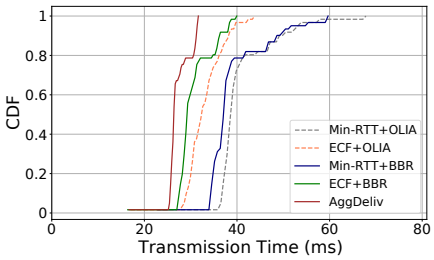

Fig. 9. Packet transmission time CDFs of AggDeliv and the comparative approaches.

transmission time, for which optimal parameters are determined by grid search.

Selection of Determinative Parameters. Our solution involves several parameters, the values of which may have a large influence on the framework's performance. Thus, we first select optimal values for them based on a grid search approach. Concretely, the congestion window affecting period T_a should commonly be tens to hundreds of milliseconds, and the factor limiting and congestion thresholds θ_γ and θ_c are both little integers. Accordingly, we pick a small collection of typical values for each parameter – (50, 100, 150, 200) ms for T_a, and (1, 2, 3, 4) for θ_γ and θ_c. Taking the average aggregate goodput as the metric, we conduct the network trace replay experiment each time with a combination of parameter values, *i.e.,* (50, 1, 1), (50, 1, 2), $\cdots$, $(200, 4, 4)$. We compare among candidate values of each parameter with the given values of the other two parameters, and select the one to achieve the highest average aggregate goodput as the optimal value. Fig. 8 shows the average aggregate goodput results of each parameter's optimal value compared with other candidate values. Accordingly, we set the parameters T_a, θ_γ, and θ_c to be 100 ms, 3, 2.

Improvement at Aggregate Goodput. We next explore how much improvement at aggregate goodput AggDeliv can achieve in the network trace replay experiment. Accordingly, we monitor the well-chosen 5 test items' aggregate goodput variations every second during the 10-minute packet scheduling period designated in §II-B, of which the results are plotted in Fig. 8. We can see that our scheduling scheme and congestion control algorithm both help improve the aggregate goodput. Despite fluctuations, the AggDeliv solution keeps performing best among all comparative approaches, where the overall improvement is $1.1\times \sim 3.2\times$. This is largely due to the adaptivity of AggDeliv's delay and loss aware solution to the heterogeneous and dynamic wireless networks.

Reduction at Packet Transmission Time. We are also concerned with how long a packet can be successfully transmitted during the network trace replay experiment. To quantify this, we show in Fig. 9 the cumulative distribution functions (CDFs) of the 5 test items' packet transmission time during all extracted trace periods. Obviously, the default MPQUIC solution ("Min-RTT + OLIA") exhibits the highest tail transmission time among the test items, and our solution is more effective in performance optimization compared with the best available scheduling and congestion control combination ("ECF + BBR"). The average packet transmission time reduction by adopting AggDeliv is 15.3% on average and at least 11.2%. This mainly comes from our design's ability to closely follow the network condition variations.

D. Performance on Live Video Delivery

We finally study AggDeliv's performance especially on live video delivery, comparing it with alternative approaches acting in the wild.

Streaming-Video Bitrate. We conduct video delivery with AggDeliv and baselines including two comparative approaches as well as transmission through two single network interfaces (WiFi and LTE). All these approaches are applied to the transmission of the five 10-minute pretreated 4K videos from the smartphone to the server in both static and walking patterns (as described in §IV-B), each of which is run for 10 times across a day to ensure the performance stability. We depict the average, maximum, and minimum streaming-video bitrates for these tests in Fig. 10, categorized by mobility pattern ("Static" or "Walking"). The overall performance of the latter category is inferior to that of the former category due to the degradation of the network conditions. Importantly, the bitrate results of AggDeliv are always higher than those of the comparative approaches (18.4% and 26.1% higher on average for the two patterns, respectively), which suggests our framework's high efficiency and robustness in practical use.

Effectiveness of Frame Coding. To explore whether our lightweight video frame coding approach is effective to maintain a high video quality during video transmission, we contrast it with two alternative approaches – 1) directly transmitting raw (uncompressed) video frames, and 2) coding the frames with H.264 [17] and conducting MP-DASH streaming [35]. We leverage the structural similarity index (SSIM) to quantify the video quality (> 0.99 as excellent, $0.95 \sim 0.99$ as good, and $0.88 \sim 0.95$ as fair [36]). We still adopt the 5 pretreated 4K videos for testing, and plot the results of three video transmission approaches in Fig. 11, categorized by a combination of mobility pattern and video richness. We can observe that AggDeliv achieves the highest SSIM among the three approaches, and it provides at least good video quality (SSIM > 0.95) for all categories, which well demonstrates the effectiveness of our frame coding design.

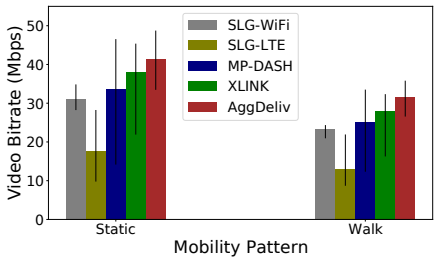

Fig. 10. Video bitrates achieved with AggDeliv and the baselines in two mobility patterns.

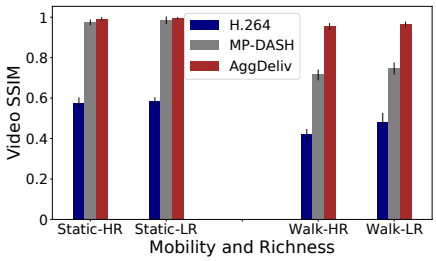

Fig. 11. Video quality comparison among different transmission approaches.

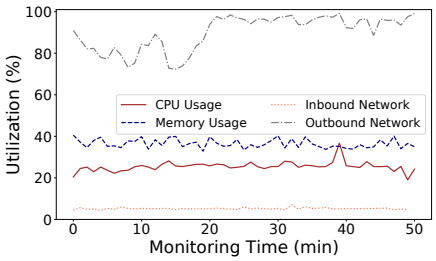

Fig. 12. AggDeliv's resource utilization during the video delivery process.

Resource Usages. Given that the average and difference calculation in the frame coding mechanism may bring a quantity of resource consumption to the mobile client, we finally monitor our Android smartphone's resource usages (the configuration is listed in §IV-B) when conducting live delivery of the five 10-minute pretreated 4K videos successively in the walking pattern. We measure utilization variations of the smartphone's CPU[1], memory, and overall inbound and outbound network traffic of both WiFi and LTE interfaces. As shown in Fig. 12, except for full use of the outbound network bandwidth, the utilization ratios of CPU, memory, and inbound network bandwidth keep low and steady confronted with intensive frame transmission, which reflects that a mobile device's resource is enough to support the delivery of 4K videos via multiple wireless networks.

V. Related Work

We discuss closely related work in the following two areas.

Multi-Path Data Transmission. Packet scheduling and congestion control are two factors affecting transmission performance. In addition to the afore-described packet scheduling schemes [4], [5], [9], [10], there are a few more similar studies. To achieve in-order receiving, DEMS [37] and STMS [11] decouple subflows by splitting data chunks and estimating the paths' gap in packet sequence number, respectively. Musher [38] addresses the limitations of penalization in wireless networks. However, these schemes that rely on network estimation suffer from wireless dynamics. There are generally two types of multi-path congestion control algorithms. Packet-loss-based algorithms (*e.g.,* LIA [15], OLIA [16], BALIA [39]) couple congestion windows of all subflows to avoid unfairness, which however often misjudge the packet loss types due to network traffic bursts. Delay-based algorithms (*e.g.,* wVegas [40], DAIMD [41]) that deem RTT as a congestion indicator are conducive to throughput improvement but perform unwell in the network with high packet loss rate. In contrast, we relate multi-path packet scheduling to congestion control optimization, which fits heterogeneous and dynamic wireless networks.

Video Streaming over Wireless Networks. Video streaming is increasingly concerned by the academia in recent years. A number of studies focus on understanding and improving the performance of rate adaptation algorithms. Most of them (*e.g.,* Pensieve [42], NAS [43], DDS [44]) only consider video-on-demand (VoD) services in which entire videos are prepared at the server for downloading. Some other papers (*e.g.,* piStream [45], OnRL [46]) particularly concern the wireless network's unreliability and throughput fluctuation in the design of video transmission. However, all the above work does not study 4K and beyond video streaming, which is difficult to be coded and transmitted in real time due to the large size. Jigsaw [19] and Rubiks [47] design layered coding approaches to minimize network traffic for 4K and 360-degree videos respectively, but they have not comprehensively considered the coding and transmission costs. Besides, MP-DASH [35] schedules streaming video traffic over MPTCP, which however solely exploits video information from the client regardless of path heterogeneity. XLINK [8] particularly applies MPQUIC to optimize short video streaming. Unlike these previous studies, our work provides efficient and robust live video streaming over multiple wireless links with proper packet scheduling and frame coding mechanisms.

[1]We currently only adopt CPU to execute the AggDeliv framework given that not all the mobile devices are equipped with GPU.

VI. Conclusion

This paper presents AggDeliv, a framework that provides efficient and robust multi-path data transmission for mobile live video delivery. We first reveal by real-world measurements the inefficiency of the state-of-the-art multi-path packet scheduling schemes and congestion control algorithms for heterogeneous and dynamic wireless links. Guided by the observations, we design enabling schemes of $cwnd$-based probabilistic packet allocation, wireless-oriented delay-loss-aware congestion control, and lightweight network-adaptive video transmission mechanisms. We put these techniques together to implement the AggDeliv system, and demonstrate its high aggregate goodput and low transmission latency for UHD video delivery with both trace-driven evaluations and experiments in the wild.

Acknowledgment

This research is supported in part by the National Natural Science Foundation of China under grants 62102224 and 62302253, the CCF-Tencent Open Fund under grant RAGR20210133, and the Beijing Natural Science Foundation under grant 4222026.

REFERENCES

[1] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proceedings of ACM SIGCOMM*, 2020, pp. 479–494.

[2] Y. Li, H. Lin, Z. Li, Y. Liu, F. Qian, L. Gong, X. Xin, and T. Xu, "A nationwide study on cellular reliability: measurement, analysis, and enhancements," in *Proceedings of ACM SIGCOMM*, 2021, pp. 597–609.

[3] C. Raiciu, M. J. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," RFC 6356, Oct. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6356

[4] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proceedings of ACM CoNEXT*, 2017, pp. 160–166.

[5] Q. De Coninck, F. Michel, M. Piraux, F. Rochet, T. Given-Wilson, A. Legay, O. Pereira, and O. Bonaventure, "Pluginizing QUIC," in *Proceedings of ACM SIGCOMM*, 2019, pp. 59–74.

[6] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath extension for QUIC," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-05, Jul. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/05/

[7] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proceedings of USENIX NSDI*, 2012, pp. 399–412.

[8] Z. Zheng, Y. Ma, Y. Liu, F. Yang, Z. Li, Y. Zhang, J. Zhang, W. Shi, W. Chen, D. Li *et al.*, "XLINK: QoE-driven multi-path QUIC transport in large-scale video services," in *Proceedings of ACM SIGCOMM*, 2021, pp. 418–432.

[9] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proceedings of IFIP Networking*, 2016, pp. 431–439.

[10] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of ACM CoNEXT*, 2017, pp. 147–159.

[11] H. Shi, Y. Cui, X. Wang, Y. Hu, M. Dai, F. Wang, and K. Zheng, "STMS: Improving MPTCP throughput under heterogeneous networks," in *Proceedings of USENIX ATC*, 2018, pp. 719–730.

[12] Y. Guo, F. Qian, Q. A. Chen, Z. M. Mao, and S. Sen, "Understanding on-device bufferbloat for cellular upload," in *Proceedings of IMC*, 2016, pp. 303–317.

[13] Y.-C. Chen and D. Towsley, "On bufferbloat and delay analysis of multipath TCP in wireless networks," in *Proceedings of IFIP Networking*, 2014, pp. 1–9.

[14] B.-H. Oh and J. Lee, "Constraint-based proactive scheduling for MPTCP in wireless networks," *Computer Networks*, vol. 91, pp. 548–563, 2015.

[15] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," Tech. Rep., 2011.

[16] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1651–1665, 2013.

[17] (2021) H.264: Advanced video coding for generic audiovisual services. https://www.itu.int/rec/T-REC-H.264-202108-I/en.

[18] (2021) H.265: High efficiency video coding. https://www.itu.int/rec/T-REC-H.265-202108-I/en.

[19] G. Baig, J. He, M. A. Qureshi, L. Qiu, G. Chen, P. Chen, and Y. Hu, "Jigsaw: Robust live 4K video streaming," in *Proceedings of ACM MobiCom*, 2019, pp. 1–16.

[20] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC transport protocol: Design and internet-scale deployment," in *Proceedings of ACM SIGCOMM*, 2017, pp. 183–196.

[21] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.

[22] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[23] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[24] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[25] (2023) Open-source MPQUIC project. http://github.com/qdeconinck/mp-quic.

[26] (2023) iPerf - The ultimate speed test tool for TCP, UDP and SCTP. https://iperf.fr/.

[27] (2023) Linux advanced routing and traffic control. http://lartc.org/howto/.

[28] (2023) Xiph.org Video Test Media [derf's collection]. https://media.xiph.org/video/derf/.

[29] (2023) FFmpeg. http://ffmpeg.org/.

[30] C.-M. Chang, C.-H. Hsu, C.-F. Hsu, and K.-T. Chen, "Performance measurements of virtual reality systems: Quantifying the timing and positioning accuracy," in *Proceedings of ACM MM*, 2016, pp. 655–659.

[31] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol," in *Proceedings of USENIX NSDI*, 2018, pp. 267–282.

[32] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with Festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, 2014.

[33] (2006) ISO/IEC 13818-7:2006 Information technology — Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC). https://www.iso.org/standard/43345.html.

[34] (2023) gRPC. https://www.grpc.io/.

[35] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH: Adaptive video streaming over preference-aware multipath," in *Proceedings of ACM CoNEXT*, 2016, pp. 129–143.

[36] A.-N. Moldovan and C. H. Muntean, "QoE-aware video resolution thresholds computation for adaptive multimedia," in *Proceedings of IEEE BMSB*, 2017, pp. 1–6.

[37] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proceedings of ACM MobiCom*, 2017, pp. 141–153.

[38] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer, "Musher: An agile multipath-TCP scheduler for dual-band 802.11 ad/ac wireless LANs," in *Proceedings of ACM MobiCom*, 2019, pp. 1–16.

[39] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 596–609, 2016.

[40] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proceedings of IEEE ICNP*, 2012, pp. 1–10.

[41] R. Gonzalez, J. Pradilla, M. Esteve, and C. E. Palau, "Hybrid delay-based congestion control for multipath TCP," in *Proceedings of IEEE MELECON*, 2016, pp. 1–6.

[42] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of ACM SIGCOMM*, 2017, pp. 197–210.

[43] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *Proceedings of USENIX OSDI*, 2018, pp. 645–661.

[44] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proceedings of ACM SIGCOMM*, 2020, pp. 557–570.

[45] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "piStream: Physical layer informed adaptive video streaming over LTE," in *Proceedings of ACM MobiCom*, 2015, pp. 413–425.

[46] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "OnRL: Improving mobile video telephony via online reinforcement learning," in *Proceedings of ACM MobiCom*, 2020, pp. 1–14.

[47] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proceedings of ACM MobiSys*, 2018, pp. 482–494.