OLMS: A Flexible Online Learning Multi-Path Scheduling Framework

Kechao Cai, *Member, IEEE*, Zhuoyue Chen, Jinbei Zhang, *Member, IEEE*, and John C. S. Lui, *Fellow, IEEE*

***Abstract*—Over the past decade, there has been a tremendous surge in the inter-connectivity among hosts in networks. Many multi-path transport protocols, such as MPTCP, MPQUIC, and MPRDMA, have emerged to facilitate multi-path data transmissions between pairs of hosts. However, existing packet schedulers in these protocols are quite limited as they neglect the stochastic nature inherent in heterogeneous paths, such as, round-trip time and available bandwidth. Moreover, users have diverse requirements; for instance, some prioritize low latency, while others consistently seek to achieve high bandwidth. In this paper, we propose a flexible Online Learning Multi-path Scheduling (OLMS) framework to schedule packets to multiple paths and meet various user-defined requirements by learning the dynamic characteristics of paths in various applications. Specifically, we consider two types of applications, which are 1) *maxRTT constrained* and 2) *bandwidth constrained*, and use OLMS to schedule packets to satisfy the distinct user-defined requirements. Our theoretical analysis demonstrates that OLMS achieves guarantees with *sublinear* regret and *sublinear* violation. Furthermore, we implement a prototype of OLMS in MPQUIC and conduct experiments across different scenarios. Our experiments on Mininet show that OLMS enables an 8.42%–18.71% increase in bandwidth utilization in the maxRTT constrained application and negligible violations of user-defined requirements in both applications compared to other schedulers. Additionally, OLMS reduces flow completion times by 4.22%–10.26% compared to other schedulers, all without incurring large overhead.**

***Index Terms*—Dynamic scheduling, multi-armed bandit problem, multi-path data transmission protocols, online learning.**

I. Introduction

NETWORKS are becoming increasingly dense with strong inter connectivity among hosts. A recent survey on routing policies in wide area networks (WANs) suggests that the Internet topology has evolved from a hierarchical topology to an increasingly meshed network topology [1], which implies a much higher number of paths connecting between two hosts in WANs.

To take advantage of the redundant paths, several multi-path transport protocols, such as MPTCP [2], MPQUIC [3], MPRDMA [4], have emerged in recent years. This development is driven by the limitations of classical *single-path* transport protocols, which lack resilience to path failures and struggle to fully exploit multi-path resources. These multi-path transport protocols are specifically designed to facilitate multi-path data transmission for devices equipped with multiple network interfaces and have been adapted to manage traffic flows within data center networks [4], [5], [6], [7]. The core functionality of multi-path transport protocols is to transfer data traffic across multiple paths from one end-host to another. In scenarios where a path becomes congested, these protocols dynamically redirect data traffic from the most congested path to those with lower congestion levels. However, the effectiveness of multi-path transport protocols is significantly impacted by the embedded multi-path scheduler, as it determines the path for transmitting packets. Specifically, the default scheduler in MPTCP and MPQUIC, known as minRTT, sends packets along the path with the lowest estimated RTT (round-trip time) until the congestion window of that path is full [2]. Then it shifts packets to the path with the next higher RTT. However, as reported in [8], minRTT suffers from low bandwidth utilization, particularly on heterogeneous paths.

While existing schedulers mainly focus on exploiting the utilization of multiple paths [9], [10], [11], they face new challenges in networks with fluctuating bandwidth and latency. In such environments, different transmission paths will experience different fluctuations in path characteristics, such as round-trip time or available bandwidth. Ignoring these changing path characteristics and naively scheduling paths can degrade the performance of applications. For example, as suggested in [12], inappropriate scheduling to paths with high latencies would adversely affect a significant fraction of requests in large-scale distributed systems. Therefore, it is critical to consider the stochastic path characteristics for better multi-path scheduling. However, scheduling an optimal path for a packet from a set of candidate paths with fluctuating characteristics is challenging. On the one hand, presetting an optimal path without any prior knowledge of the path characteristics is not feasible. For instance, in the initialization of a multi-path connection, there is no historical information on path characteristics available for

Received 13 March 2024; revised 25 January 2025; accepted 19 February 2025. Date of publication 3 March 2025; date of current version 25 April 2025. The work of Kechao Cai and Jinbei Zhang was supported in part by the National Key R&D Program of China under Grant 2022YFB2902700, in part by NSF China under Grant 62202508, Grant 62071501, and Grant 62471505, and in part by Shenzhen Science and Technology Program under Grant 20220817094427001, Grant JCYJ20220818102011023, and Grant ZDSYS20210623091807023. The work of John C.S. Lui was supported by RGC GRF under Grant 14202923. Recommended for acceptance by Dr. Yuan Wu. *(Corresponding author: Jinbei Zhang.)*

Kechao Cai, Zhuoyue Chen, and Jinbei Zhang are with the School of Electronics and Communication Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen 510275, China (e-mail: caikch3@mail.sysu.edu.cn; chenzhy225@mail2.sysu.edu.cn; zhjinbei@mail.sysu.edu.cn).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNSE.2025.3546957, provided by the authors.

Digital Object Identifier 10.1109/TNSE.2025.3546957

a host to make appropriate scheduling decisions. On the other hand, the constantly changing path characteristics add complexity to multi-path scheduling, making it challenging to identify proper paths that offer long-term benefits for data packets. Moreover, different applications impose different user-defined requirements on multi-path scheduling. For example, real-time video streaming services would require the latency to be under a certain threshold [13], as high latency would deteriorate the quality of video services.

To tackle these challenges, we approach the multi-path scheduling problem as an online learning problem. In this paradigm, the multi-path scheduler is not required to possess prior statistics of path characteristics (e.g., available bandwidth and overall latency) but learns these statistics and optimizes multi-path scheduling in an online fashion. In particular, we propose a novel online learning framework derived from the Multi-Armed Bandit (MAB) model, which enables network performance optimization with partial observations of path characteristics throughout the learning process. Different from previous MAB work [14], [15], the main purpose of our online learning framework is not only to find the optimal set of arms (paths) to minimize the *regret* (which is the performance gap between the optimal policy and our multi-path scheduling algorithm during the learning process), but also to try to *satisfy the user-defined requirements* in applications. This is important for many real-world applications. For example, it is crucial to schedule packets to paths with high available bandwidth while simultaneously bounding the average latency in latency-sensitive applications [16], [17]. Therefore, beyond merely minimizing the regret, our learning algorithm also considers *violation*, measuring how much the learning algorithm breaches the user-defined requirements over time. Without these constraints, the path scheduling algorithm could go wild without meeting any requirements in applications.

In this work, we propose a flexible Online Learning Multi-path Scheduling (OLMS) framework. Our framework integrates constraints into the MAB model, facilitating scheduling packets to multiple paths to meet user-defined requirements in various applications. We develop *a novel sampling technique based on Thompson sampling* [18] in OLMS to estimate fluctuating path characteristics without assuming any prior distributions. We then use the generated posterior samples as the estimates for path characteristics to schedule packets to the optimal path under the user-defined requirements, and update the distributions of path characteristics of the scheduled path when receiving new measurements. Furthermore, we develop a novel path characteristic monitor to detect abrupt network changes and provide more timely estimates of path characteristics to enhance the adaptability of OLMS.

To demonstrate the flexibility of OLMS, we consider two different applications, namely, *maxRTT constrained* and *bandwidth constrained* multi-path scheduling applications to cover most common requirements in real-world applications. For each application, we specialize different components in the OLMS framework, and we prove that our specialized algorithm achieves both *sublinear regret* and *sublinear violation*. We also implement a prototype of our OLMS framework in MPQUIC and conduct extensive experiments on Mininet. Our experiments show that OLMS increases the bandwidth utilization by 8.42%–18.71% in the *maxRTT constrained* application, and incurs negligible violations in both applications compared to other schedulers. In addition, OLMS reduces flow completion times by 4.22%–10.26% compared to other schedulers, all without incurring any significant overhead.

In summary, we make the following contributions. 1) To the best of our knowledge, we are the first to present a flexible *online learning multi-path scheduling framework with user-defined requirements* for multi-path transport protocols. 2) We design a general multi-path scheduling algorithm that performs well while satisfying the diverse user-defined requirements in real-world applications. 3) We also prove that our algorithm achieves sublinear regret and sublinear violation. 4) We implement a prototype of our framework in MPQUIC and demonstrate its effectiveness in multi-path scheduling across different scenarios, without incurring large overhead.

II. Framework Design

In this section, we give the definitions of time slots and path characteristics, and present our Online Learning Multi-path Scheduling (OLMS) framework design. Then we define the general regret and violation to measure the performance of our framework.

A. Time Slots and Path Characteristics

Throughout a multi-path data transmission session, the scheduler at the sender would constantly find available paths to transmit packets to the receiver. We divide the total data transmission time into a series of time slots where each time slot corresponds to the duration of a single packet transmission. These time slots serve as the clocked intervals in our framework. At each time slot, the scheduler schedules one path from the K paths for each packet transmission. This concept of "time slots" is similar to the "Monitor Intervals (MIs)" in PCC Vivace [19] and PCC Proteus [20], where network measurements and operations are conducted within these time intervals. From now on, we refer to the t-th time slot as time t.

To capture the characteristics of various paths in multi-path data transmissions, we follow the salient works (BBR [21], PCC [22], PCC Vivace [19], and PCC Proteus [20]) on congestion control, and model a path using two basic characteristics: round-trip time (R_{rtt}), and available bandwidth (B_{bw}). Let $\mathcal{K} = \{1, \ldots, K\}$ denote a set of K candidate paths from the sender to the receiver. The R_{rtt} of a path represents the end-to-end latency caused by data transmission, propagation, and network queuing; the B_{bw} of a path represents the maximum rate at which the sender can transmit along this path to its receiver.

B. OLMS Framework

We implement our Online Learning Multi-path Scheduling (OLMS) framework to learn the two characteristics of multiple paths and schedule packets to the proper paths. Our implementation is built upon MPQUIC, and is deployed at the sender in order to control the multi-path scheduling. Fig. 1 gives an overview of our framework design. Specifically, the implementation contains

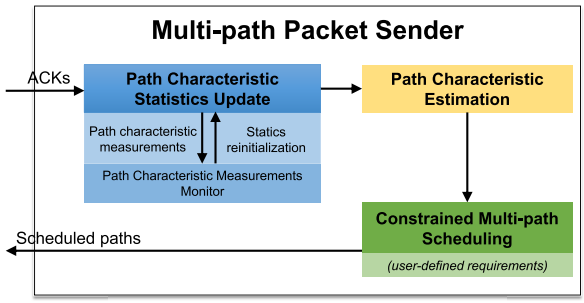


Fig. 1. An overview of OLMS framework.

three components: (i) *Path Characteristic Statistics Update*, (ii) *Path Characteristic Estimation*, (iii) *Constrained Multi-path Scheduling*.

In *Path Characteristic Statistics Update*, the sender receives acknowledgments (ACKs) from the receiver via the scheduled paths and obtains measurements of path characteristics at time t. Using these measurements, the sender updates the posterior distribution of path characteristics for the scheduled path. At time $t+1$, the sender will generate new estimates of the path characteristics via the *Path Characteristic Estimation* component. Then the sender transmits the packet along the path scheduled via the *Constrained Multi-path Scheduling*. Inside the *Path Characteristic Statistics Update* component, there is a sub-component called *Path Characteristic Measurements Monitor*. In this sub-component, the sender can detect abrupt network changes based on the measurements and posterior distribution of path characteristics. Upon detecting a network change, the sender will reinitialize the path characteristic statistics.

In *Path Characteristic Estimation*, at time $t \geq 1$, the sender generates path characteristics estimates of the K candidate paths from the updated posterior distributions using the Thompson Sampling method. Specifically, we model the *unknown* prior distributions of path characteristics of the K candidate paths by generalizing the Bernoulli distributions (see details in § III). This approach does not require knowledge of prior distributions, which differs from the classical sampling techniques. These estimates are then input into the optimization module in the *Constrained Multi-path Scheduling*, where the sender schedules the packet to the optimal path $p_t \in \{1, \ldots, K\}$ from the K candidate paths that optimizes the utility (e.g., delay, bandwidth) and satisfies the *user-defined requirements* in applications. The sender then transmits the packet along the scheduled optimal path.

In summary, our OLMS framework measures the path characteristics, updates the statistics, estimates the characteristics, transmits data along the scheduled optimal path, and obtains new path characteristic measurements in an iterative fashion. This closed control loop enables flexible and adaptive multi-path packet scheduling.

C. General Regret and Violation

In our OLMS framework, different applications may tailor their implementations of the three components to align with their unique requirements. Despite the diversity of implementations across different applications, we use two common metrics, *regret* and *violation*, to measure their performance.

Let $U(p_t)$ be the general utility function of path p_t. The regret of a multi-path scheduling algorithm measures the cumulative loss of utility of the scheduled path compared against the utility of the optimal path scheduling algorithm[1] which knows all path characteristics and can consistently schedule the optimal path at each time slot. Define the utility of the optimal path scheduling algorithm at time t as $\texttt{OPT}$. Specifically, in the context of a maximization problem, compared to the maximal optimal utility $\texttt{OPT}_{\max}$, the general regret for a multi-path scheduling algorithm up to time T is defined as follows,

$$\text{Reg}_{\max}(T) = \mathbb{E}\left[\sum_{t=1}^{T} (\text{OPT}_{\max} - U(p_t))^+\right], \quad (1)$$

where the expectation is taken over the randomness of the algorithm in scheduling the paths and $(\cdot)^+ = \max(\cdot, 0)$. Here we only take account of the non-negative values in calculating the regret as suboptimal performance arises only when $\texttt{OPT}_{\max}$ is greater than $U(p_t)$. Conversely, for a minimization problem, the minimal optimal utility is $\texttt{OPT}_{\min}$ and the general regret can be defined as

$$\text{Reg}_{\min}(T) = \mathbb{E}\left[\sum_{t=1}^{T} (U(p_t) - \text{OPT}_{\min})^+\right]. \quad (2)$$

Note that the smaller the regret, the closer the algorithm approaches the optimal algorithm. Thus, the regret should be as small as possible.

Given that the sender can only use the estimates of path characteristics to schedule paths, the actual path characteristics of the scheduled path may initially violate the user-defined requirements. This is particularly evident when there is little information about the paths during the connection setup. Let the general constraint function $G(p_t)$ $(G(p_t) \leq 0)$ represent the user-defined requirement on path p_t for $p_t \in \mathcal{K}$. A positive violation $G(p_t)$ will be incurred if the constraint $G(p_t) \leq 0$ is violated. Thus, we define the general overall violation of the constraint of a path scheduling algorithm at time T as follows:

$$\text{Vio}(T) = \mathbb{E}\left[\sum_{t=1}^{T} G(p_t)^+\right], \quad (3)$$

where the expectation is taken over the randomness of the algorithm in scheduling the paths. Here a small violation means that the path scheduled by the algorithm seldom violates the constraint. Thus, the violation should be as small as possible. Note that the violation defined in (3) accumulates the *step-wise constraint violation* over T time slots. This violation is different from violations defined on the *long-term constraint satisfaction*, i.e., $(\sum_{t=1}^{T} G(p_t))^+$ in [23] and [24], where the feasible solutions to $G(p_t) \leq 0$ can cancel out the effects of violated constraints.

[1]Note that the optimal multi-path scheduling algorithm may not be realizable in practice because it requires the full knowledge of the stochastic nature of the path characteristics.

III. Algorithm Design

In this section, we first present the general path modeling and elaborate on the multi-path scheduling algorithm in the OLMS framework. Then we provide a detailed description of the sub-component *Path Characteristic Measurements Monitor*.

A. Path Modeling

Let the random variable r_i^t characterize the round-trip time (R_{rtt}) of path i at time $t \geq 1$ and $r_i = \mathbb{E}[r_i^t]$ be the *unknown* mean of r_i^t for $i \in \mathcal{K}$. Let $\hat{r}_i^t$ be the estimate of r_i^t at time t. For the R_{rtt} of the K paths, we have the vectors[2] $\boldsymbol{r}_t = (r_1^t, \ldots, r_K^t)$, $\boldsymbol{r} = (r_1, \ldots, r_K)$, and $\hat{\boldsymbol{r}}_t = (\hat{r}_1^t, \ldots, \hat{r}_K^t)$. Let the random variable b_i^t denote the bandwidth (B_{bw}) of path i at time t and $b_i = \mathbb{E}[b_i^t]$ denote the *unknown* mean of b_i^t for $i \in \mathcal{K}$. Denote $\hat{b}_i^t$ as the estimate of b_i^t at time t. Similarly, for the B_{bw} of the K paths, we have the vectors $\boldsymbol{b}_t = (b_1^t, \ldots, b_K^t)$, $\boldsymbol{b} = (b_1, \ldots, b_K)$, and $\hat{\boldsymbol{b}}_t = (\hat{b}_1^t, \ldots, \hat{b}_K^t)$.

For mathematical tractability, we assume that the both path characteristics, R_{rtt} and B_{bw} of the K paths, are independent of each other. Such independence among the path characteristics is also suggested in BBR [21] and PCC Vivace [19], where the path characteristics are measured with different uncorrelated functions. Without loss of generality, we normalize these characteristics to [0,1] with min-max scaling, i.e., $r_i^t \in [0, 1]$, and $b_i^t \in [0, 1]$ for $i \in \mathcal{K}, t \geq 1$.

At each time t, we consider scheduling a packet to the optimal path in the K candidate paths. Here we point out that a good multi-path scheduling algorithm should find the optimal path by taking the random nature of the path characteristics into consideration. In other words, the algorithm should continuously explore the optimal path and adjust the probabilities of scheduling the K paths by estimating the path characteristics. Otherwise, the regret and violation of the algorithm would be linear in time T. A simple example is the uniform random algorithm where each path is scheduled with equal probability. This algorithm neglects the heterogeneity of the path characteristics. As a result, it incurs both *linear* regret and *linear* violation as there can be a constant loss in the utility $U(p_t)$ compared to $\texttt{OPT}_{\max}$ or $\texttt{OPT}_{\min}$ at each time t, where p_t represents the scheduled path at time t. Let $\boldsymbol{v}_t = (v_1^t, \ldots, v_K^t)$ be the probabilistic path scheduling vector for the K paths at time t, where $v_i^t \in [0, 1]$ is the probability of scheduling a packet to path i at time t for $i \in \mathcal{K}$. We have $\sum_{i=1}^{K} v_i^t = L$ $(1 \leq L < K)$ as we aim to identify an optimal set of L paths and schedule packets to a path within this set at each time t. Let $\mathbf{1}$ be the K-dimensional vector whose elements are all ones. Let $N_i^{B,t}$ and $N_i^{R,t}$ be the number of times that b_i^t and r_i^t are measured from path i at the beginning of time t, i.e., $N_i^{B,t} = \sum_{\tau<t} \mathbb{1}(i \in p_\tau)$, $N_i^{R,t} = \sum_{\tau<t} \mathbb{1}(i \in p_\tau)$ and $N_i^{B,1} = 0$, $N_i^{R,1} = 0$. Here $\mathbb{1}(\mathcal{E})$ is an indicator function. Let $\chi = \{\boldsymbol{v} \in \mathbb{R}^K | 0 \leq v_i \leq 1, \mathbf{1}^\intercal \boldsymbol{v} = L\}$ be the set of feasible probabilistic path scheduling vectors.

[2]All vectors are column vectors.

Algorithm 1: General Algorithm Design in OLMS.

1: **Init:** $S_i^B = S_i^R = 0$, $N_i^{B,1} = N_i^{R,1} = 0$ for $i \in \mathcal{K}$
2: Establish K paths between sender and receiver.
3: **for** $t = 1, 2, \ldots, T$ **do**
▷ *Path Characteristic Estimation*:
4: **for** $i = 1, 2, \ldots, K$ **do**
5: $\hat{b}_i^t = \text{Beta}(S_i^B + 1, N_i^{B,t} - S_i^B + 1)$
6: $\hat{r}_i^t = \text{Beta}(S_i^R + 1, N_i^{R,t} - S_i^R + 1)$
▷ *Constrained Multi-path Scheduling*:
7: Solve a linear program for path scheduling vector $\boldsymbol{v}_t$.
8: Select a path p_t with $\boldsymbol{v}_t$ and send packet along p_t.
▷ *Path Characteristic Statistics Update*:
9: $b_{p_t}^t, r_{p_t}^t \leftarrow$ measurements of p_t's B_{bw}, R_{rtt}
▷ *Path Characteristic Measurements Monitor*:
10: MONITOR($b_{p_t}^t$, p_t) ▷ Bandwidth monitor
11: MONITOR($r_{p_t}^t$, p_t) ▷ RTT monitor
12: $S_{p_t}^B \leftarrow S_{p_t}^B + 1$ if Bernoulli($b_{p_t}^t$) $= 1$
13: $S_{p_t}^R \leftarrow S_{p_t}^R + 1$ if Bernoulli($r_{p_t}^t$) $= 1$
14: $N_{p_t}^{B,t+1} \leftarrow N_{p_t}^{B,t} + 1$, $N_{p_t}^{R,t+1} \leftarrow N_{p_t}^{R,t} + 1$

B. General Algorithm Design

Algorithm 1 shows the *general* design of our online learning multi-path scheduling (OLMS) framework. First, the multi-path packet sender establishes K data transmission paths with the receiver (Line 2). At each time t, via the *Path Characteristic Estimation* (Line 5-7), the sender gets the estimates of the path characteristics of the K paths. In particular, note that the probability of observing a success in a Bernoulli trial Bern(r_i) with probability r_i is exactly the expectation of the random variable r_i^t with unknown probability density function $f(r_i^t)$. That is, $\Pr(r_i^t = 1) = \int_0^1 r_i^t f(r_i^t) dr_i^t = r_i$. Hence, one can perform Bernoulli trials with the success probabilities b_i^t, and r_i^t, and use random samples from Beta distributions which are conjugate priors of Bernoulli distributions to estimate the path characteristics $\hat{b}_i^t$, and $\hat{r}_i^t$, respectively.

More specifically, let S_i^B be the number of successful Bernoulli trials with unknown probability b_i^t up to time t. Initially, we have $S_i^B = 0$ and $S_i^R = 0$ at time $t = 1$ since no path has been scheduled. Then we can give Bayesian estimates for the B_{bw} and R_{rtt} of path $i \in \mathcal{K}$, $\hat{b}_i^t$ and $\hat{r}_i^t$, by sampling from the Beta distributions at time t as shown in Line 6 and Line 7, respectively. For large $N_i^{B,t}$, according to the law of large numbers, the estimate $\hat{b}_i^t$ in Line 6 would be concentrated on the fraction $S_i^B / N_i^{B,t}$. This fraction is approximately equal to b_i and is equal to the expected successful probability of the Bernoulli trial Bernoulli(b_i^t). Similarly, the estimate $\hat{r}_i^t$ in Line 7 would be close to r_i for large $N_i^{R,t}$.

In the *Constrained Multi-path Scheduling* (Line 9-10), the sender uses the estimates as input to solve a linear program (LP) $optimize_{G(p_t) \leq 0, \boldsymbol{v}_t \in \chi} U(p_t)$ to find the path scheduling vector $\boldsymbol{v}_t$ that optimizes the utility function, and at the same time, satisfies the constraint of a given multi-path scheduling application. If the LP has no feasible solution, OLMS would fall back to default path scheduler minRTT. Then it samples a set of L paths from

Algorithm 2: Path Characteristic Measurements (B_{bw}) Monitor.

Init: q, δ, h, $c_i^B = 0$ for $i \in \mathcal{K}$
1: **function** Monitorb_i^t, i
2: **if** $N_i^{B,t} < q$ **then return**
3: $\mu_b = \frac{S_i^B+1}{N_i^{B,t}}$, $\sigma_b = \sqrt{\frac{(S_i^B+1)(N_i^{B,t}-S_i^B+1)}{(N_i^{B,t}+3)(N_i^{B,t}+2)^2}}$
4: **if** $|b_i^t - \mu_b| > \delta\sigma_b$ **then**
5: $c_i^B \leftarrow c_i^B + 1$
6: **if** $c_i^B > h$ **then**
7: $S_i^B \leftarrow 0$, $N_i^{B,t} \leftarrow 0$ ▷ Statistics reinitialization
8: $c_i^B \leftarrow 0$ ▷ Counter reset

the K candidate paths probabilistically using the path scheduling vector $\boldsymbol{v}_t$ via a randomized rounding scheme [25]. From this set, it selects a path p_t with the highest $v_t^{p_t}$ from that set. Then the sender transmits the packet along the scheduled path p_t.

Finally, in the *Path Characteristic Statistics Update* (Line 12-18), when the sender obtains new measurements of the path characteristics from the scheduled path p_t, it first examines whether the distributions of the path characteristics are shifted or not with the *Path Characteristic Measurements Monitor*. Specifically, the sender invokes the MONITOR function (detailed in Algorithm 2) for both the bandwidth and the RTT measurement of path p_t and check if the path characteristic statistics (S_i^B, $N_i^{B,t}$, S_i^R, and $N_i^{R,t}$) should be updated. Then it updates the parameters using Bernoulli trials, Bernoulli(b_i^t) and Bernoulli(r_i^t), as shown in Line 16, and Line 17 in Algorithm 1, respectively. Overall, Algorithm 1 operates in an iterative fashion to find the optimal path in different applications.

C. Path Characteristic Measurements Monitor

As the distributions of bandwidth or RTT can change due to network fluctuations, the estimates of bandwidth and RTT from the *Path Characteristic Estimation* component could be inaccurate. To address this issue, we propose the *Path Characteristic Measurements Monitor* sub-component in the *Path Characteristic Statistics Update* to detect the shift of distributions of the path characteristics. Take the *Path Characteristic Measurements Monitor* for B_{bw} in Algorithm 2 as an example. At each time t, the function MONITOR takes the bandwidth measurement (b_i^t) and the index of path (i) as the input. It first disregards the initial q measurements to avoid insufficient statistics, as shown in Line 3. Then it computes the average bandwidth μ_b and standard deviation of the bandwidth σ_b using the bandwidth statistics, as shown in Line 4. When the bandwidth measurement deviates from the average bandwidth by $\delta\sigma_b$ ($|b_i^t - \mu_b| > \delta\sigma_b$), it increments the deviation counter of bandwidth (c_i^B) by 1. Upon reaching a threshold of h such deviations, the *Path Characteristic Measurements Monitor* reinitializes the path characteristic statistics and resets the deviation counter, as shown in Line 8 and 9, respectively. The *Path Characteristic Measurements Monitor* for R_{rtt} operates in a similar manner to that for B_{bw}, except that the MONITOR function takes the R_{rtt} measurement as input and has the capability to reinitialize the R_{rtt} statistics.

IV. APPLICATIONS AND ALGORITHM ANALYSIS

In this section, we consider two types of applications which users can specify the constraints, they are 1) *maxRTT constrained*, and 2) *bandwidth constrained*multi-path scheduling applications. For each application, we specify the utility of the optimal algorithm, the utility function, and the constraint function in the *Constrained Multi-path Scheduling* component in our general algorithm design (see Algorithm 1), and provide theoretical guarantees on the specialized regret and violation.

A. MaxRTT Constrained Multi-path Scheduling

In scenarios where the underlying network paths are heterogeneous, multi-path transport protocols often perform worse than single-path transport protocols, particularly in latency-sensitive applications such as search engines and financial trading applications [26], [27]. These latency-sensitive applications can benefit from shaving off even fractions of a second of latency to improve the users' experience or boost the revenue. Moreover, maintaining consistently low latency is critical, as even a small number of delayed operations can cause a ripple effect that degrades the application performance [12]. In a multi-path connection, where packets can traverse different paths with varying round-trip times (R_{rtt}) due to path heterogeneity, the maximum R_{rtt} among these paths becomes the crucial determinant of the application's latency. By imposing a threshold on the maximum R_{rtt}, we can effectively control the end-to-end delay in multi-path transmission. This ensures that even if individual paths exhibit large latencies, the overall latency of the transmission remains within acceptable bounds, thereby enhancing the reliability and performance of the application.

In this subsection, we consider the *maxRTT constrained* application: *Scheduling a path at each time slot to maximize the available bandwidth, subject to the maximum RTT of the scheduled path is no greater than a preset threshold* C_r throughout the data transmission.

To adapt the *maxRTT constrained* application to the OLMS framework, we specify the *Constrained Multi-path Scheduling* component in Algorithm 1. Specifically, we solve the LP for the path scheduling vector $\boldsymbol{v}_t$ using the estimates of the B_{bw} and R_{rtt}. Let $\boldsymbol{C}_r = C_r\mathbf{1}$ and $\boldsymbol{x} \preceq \boldsymbol{C}_r$ if $x_i \leq C_r$ for all $i = 1, \ldots, K$, where $\boldsymbol{x} = (x_1, \ldots, x_K)$ is a K-dimensional vector. Let $\circ$ be the element-wise product operator of two vectors. Then the LP for the *maxRTT constrained* application can be expressed as follows,

$$\boldsymbol{v}_t = \underset{\boldsymbol{v}\in\mathcal{X},\, \hat{\boldsymbol{r}}_t\circ\boldsymbol{v}\preceq\boldsymbol{C}_r}{\arg\max} \left\{\boldsymbol{v}^\intercal \hat{\boldsymbol{b}}_t\right\}. \tag{4}$$

In (4), the objective is to *maximize* the available bandwidth, i.e., $\boldsymbol{v}^\intercal\hat{\boldsymbol{b}}_t$, of different paths, given the B_{bw} estimates $\hat{\boldsymbol{b}}_t$ and the R_{rtt} estimates $\hat{\boldsymbol{r}}_t$ of the paths. The constraint $\hat{\boldsymbol{r}}_t \circ \boldsymbol{v} \preceq \boldsymbol{C}_r$ is to guarantee that the R_{rtt} estimate of any scheduled path is no greater than C_r.

Now we give the specific utility function $\mathsf{OPT}_{\max}$ of the optimal path scheduling algorithm, utility function $U(p_t)$, and the constraint function $G(p_t)$ for the *maxRTT constrained* application. Given the full knowledge of the available bandwidth B_{bw}

($\boldsymbol{b}$) and round-trip time R_{rtt} ($\boldsymbol{r}$) of the K paths, the optimal algorithm $\mathsf{OPT}_{\max}$ can find the optimal path scheduling vector $\boldsymbol{v}^*$ to maximize the available bandwidth and satisfy the constraint on the maximum latency. Specifically, the $\mathsf{OPT}_{\max}$ in the *maxRTT constrained* application can be expressed as

$$\mathsf{OPT}_{\max} \equiv \max_{\boldsymbol{v}\in\chi, \boldsymbol{r}\circ\boldsymbol{v}\preceq\boldsymbol{C}_r} \{\boldsymbol{v}^\intercal \boldsymbol{b}\}, \tag{5}$$

and the optimal path scheduling vector is $\boldsymbol{v}^* = \arg\max_{\boldsymbol{v}} \mathsf{OPT}_{\max}$. In addition, the utility $U(p_t)$ in the *maxRTT constrained* application is the available bandwidth of the scheduled path at time t:

$$U(p_t) \equiv b_{p_t}. \tag{6}$$

As the maximum R_{rtt} of the scheduled path should be no greater than the preset threshold C_r, we define the constraint function $G(p_t)$ for the *maxRTT constrained* application as follows,

$$G(p_t) \equiv r_{p_t} - C_r. \tag{7}$$

Note that violation occurs when $G(p_t) \geq 0$ if r_{p_t} is greater than C_r at time t in (7). With the function definitions in (5)-(7), we can compute the regret and violation for the *maxRTT constrained* application.

At this stage, we have specified the functions in the *Constrained Multi-path Scheduling* component in Algorithm 1 and have adapted the *maxRTT constrained* multi-path scheduling to our OLMS framework.

B. Bandwidth Constrained Multi-path Scheduling

Many cellular Internet Service Providers (ISPs) currently offer plans featuring unlimited data usage but limited data rates achieved through bandwidth throttling. This strategy aims to attract cellular users who prioritize low-latency applications, such as sending chat messages or browsing text-rich websites, over high bandwidth. By offering plans tailored to users who are not sensitive to bandwidth but value low latency, ISPs can cater to a diverse range of user preferences and enhance their overall service offerings.

To improve the users' experience, it is also crucial for the ISPs to deliver data with minimum delay under the bandwidth throttling constraint. However, the deployment of multi-path transport protocol on mobile devices [28], [29] poses challenges in this regard, as data are transmitted through multiple paths instead of a single one. Traditional token-based bandwidth throttling methods, originally designed for single network interfaces, cannot guarantee minimal delay in multipath transport protocols. To achieve the minimal latency under the bandwidth throttling constraint, one can schedule paths that minimize the maximum R_{rtt} while limiting the available bandwidth. Specifically, we consider the *bandwidth constrained* application: *Scheduling a path at each time slot such that the maximum RTT of the scheduled path is minimized, with the requirement that the available bandwidth of the scheduled path is at most C_b throughout the data transmission*. Here C_b is an application specific bandwidth throttling threshold.

For the *Constrained Multi-path Scheduling* component in the *bandwidth constrained* multi-path scheduling, the corresponding path scheduling vector $\boldsymbol{v}_t$ can be expressed as:

$$\boldsymbol{v}_t = \operatorname*{argmin}_{\boldsymbol{v}\in\chi,\, \boldsymbol{v}^\intercal \hat{\boldsymbol{b}}_t \leq C_b} \left\{ \max_{i\in\mathcal{K}} \{v_i \hat{r}_i^t\} \right\}. \tag{8}$$

In (8), the objective is to *minimize* $\max_{i\in\mathcal{K}} v_i \hat{r}_i^t$, which is the maximum R_{rtt} of the scheduled path i. The constraint $\boldsymbol{v}^\intercal \hat{\boldsymbol{b}}_t \leq C_b$ is to ensure that the available bandwidth of the scheduled paths cannot exceed C_b.

Now we give the specific utility function $\mathsf{OPT}_{\min}$ of the optimal algorithm, utility function $U(p_t)$, and the constraint function $G(p_t)$ for the *bandwidth constrained* application. Given the K paths' available bandwidth $\boldsymbol{b}$ and round-trip time $\boldsymbol{r}$, the $\mathsf{OPT}_{\min}$ for the *bandwidth constrained* application can be expressed as

$$\mathsf{OPT}_{\min} \equiv \min_{\boldsymbol{v}\in\chi,\, \boldsymbol{v}^\intercal \boldsymbol{b} \leq C_b} \left\{ \max_{i\in\mathcal{K}} \{v_i r_i\} \right\}, \tag{9}$$

and the optimal path scheduling vector is $\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \mathsf{OPT}_{\min}$. In addition, the utility function for the *bandwidth constrained* multi-path scheduling is as follows,

$$U(p_t) \equiv r_{p_t}, \tag{10}$$

which is the maximum R_{rtt} of the scheduled path, i.e, p_t, at time t.

As the available bandwidth of the scheduled path is no greater than the throttling bandwidth C_b, the constraint function $G(p_t)$ in the *bandwidth constrained* application is defined as follows,

$$G(p_t) \equiv b_{p_t} - C_b. \tag{11}$$

In above, violation occurs if the bandwidth b_{p_t} of the scheduled path p_t is greater than the throttling bandwidth C_b at time t.

Using the function definitions in (9)–(11), we can compute the regret and violation for the *bandwidth constrained* application. This completes the adaptation of the *bandwidth constrained* application to the OLMS framework.

C. Theoretical Guarantee of OLMS

Our OLMS framework has some attractive theoretical properties which we state in the following theorem.

Theorem 1: Suppose that the path characteristics, R_{rtt} and B_{bw} of path i, are independent and identically distributed (i.i.d.) over time for $i \in \mathcal{K}$, with the unknown mean $r_i = \mathbb{E}[r_i^t]$ and $b_i = \mathbb{E}[b_i^t]$, and assume that R_{rtt} and B_{bw} are independent across different paths. By adapting the two multi-path scheduling applications, namely the maxRTT constrained multi-path scheduling, and bandwidth constrained multi-path scheduling, to the OLMS framework, for each application, we can guarantee that the upper bound of the regrets in (1) and (2) are

$$\begin{aligned} \text{Reg}_{\min}(T) &\leq O\left(\sqrt{KT\log T}\right), \\ \text{Reg}_{\max}(T) &\leq O\left(\sqrt{KT\log T}\right), \end{aligned} \tag{12}$$

respectively, and the upper bound of the violation in (3) is

$$\text{Vio}(T) \leq O\left(\sqrt{KT\log T}\right). \tag{13}$$

For the two applications of the OLMS framework, as we have different utility functions and constraint functions, the analyses of regrets and violations of the applications are also different. Thus, we prove Theorem 1 separately for each application. Notably, OLMS remains effective, albeit with a less tight bound, when the path characteristics are dependent [30]. We refer interested readers to the supplementary material for the details of the proofs of the above theorem.

V. Implementation

We implement OLMS in an MPQUIC [3]-enabled sender based on *quic-go*. The implementation of OLMS[3] consists of four key components, as described below:

Negotiation mechanism; The negotiation mechanism component enables an application to communicate its requirements to the server/sender during the connection establishment phase. During connection initiation, the application specifies its requirements (e.g., maxRTT constrained and bandwidth constrained) as part of the setup parameters. These requirements are encoded in a custom packet compatible with the MPQUIC transport protocol. The sender integrates the received requirements into the OLMS framework to configure the linear programming solver. This ensures that the OLMS framework is properly configured with user-defined requirements, enabling tailored scheduling decisions and improved performance.

Constrained multi-path scheduling: The Constrained Multi-Path Scheduling component selects optimal paths for packet transmission while satisfying user-defined requirements. During each transmission, the Path Characteristic Estimation component (described in the next paragraph) updates path characteristics (e.g., RTT and bandwidth), which are then used as input for a linear programming (LP) solver. The scheduling vector $\boldsymbol{v_t}$ is determined by solving the LP problem using a standard technique [31]. Dependent rounding technique is then applied to convert the probabilistic scheduling vector into deterministic path selections. Finally, one path is randomly chosen for packet transmission. This approach balances the exploration of multiple paths with the exploitation of high-utility paths, ensuring robust and efficient scheduling.

Path characteristic estimation: The Path Characteristic Estimation component provides accurate and dynamic estimates of path characteristics. It uses posterior Beta distributions to model the characteristics and generate estimates by sampling from the continuously updated distributions (see Algorithm 1). Upon receiving ACKs, bandwidth and RTT measurements are computed using the congestion window and the smoothed RTT of each path. These measurements are then used to update the posterior distributions, ensuring the estimates remain accurate and adapt to current network conditions.

Path characteristic statistics update: The Path Characteristic Statistics Update component collects measurements and monitors them for any abrupt changes. When ACKs are received, the smoothed RTT of each path is retrieved using MPQUIC's internal function `pth.rttStats.SmoothedRTT()`, and the congestion window is obtained with function `pth.sentPacketHandler.GetCongestionWindow()`. Bandwidth for each path is then calculated as the ratio of the congestion window size to the smoothed RTT. To detect abrupt changes in path characteristics, we implement the MONITOR() function in the Path Characteristic Measurements Monitor to track deviations of measurements from their expected distributions. Path statistics are reinitialized only if these deviations persist for h consecutive monitoring intervals, ensuring robust adaptation to network changes while reducing false alarms (see Algorithm 2).

[3]Our implementation of OLMS is open-source and available at https://github.com/MLCL-SYSU/OLMS-MPQUIC.git.

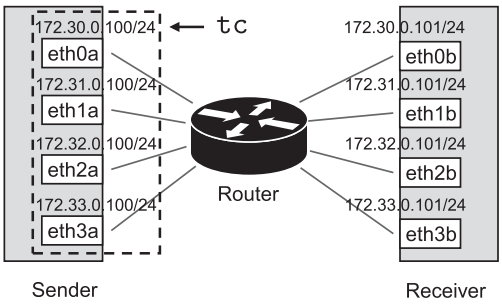


Fig. 2. Network setup with a multi-path packet sender and receiver.

VI. Experiments

In this section, we conduct experiments on the *maxRTT constrained* and *bandwidth constrained*multi-path scheduling applications under a controlled network environment. Specifically, we employ the congestion control algorithm OLIA [32] at the sender, and compare the performance of OLMS framework against the different schedulers, including Round Robin (RR), minRTT [33], ECF [34], BLEST [35], and Peekaboo [36] for each application.

A. Network Setup

1) Network Topology: To evaluate the performance of the two applications in our OLMS framework, we set up a network with the topology shown in Fig. 2. We deploy OLMS in Mininet 2.2.2 [37] on Ubuntu 18.04 and use the traffic control tool `tc`[4] to conduct controlled experiments for the study of the efficacy of different schedulers under various path characteristic settings. This experimental setup facilitates a comprehensive evaluation of the OLMS framework's performance across different network configurations and conditions.

As shown in Fig. 2, each Ethernet port (eth0a-eth3a and eth0b-eth3b) is configured with a unique IP address with a network mask, and all ports are connected to an 8-port Gigabit Ethernet router. If all Ethernet ports on the sender are configured to communicate directly with those on the client, an MPQUIC connection can establish up to 16 distinct paths between the server and the client. Modern smart devices, leveraging advancements such as the DSDA (Dual SIM Dual Active) technique in

[4]https://man7.org/linux/man-pages/man8/tc.8.html

5G networks [38] and the DBDC (Dual-Band Dual-Concurrent) technique in Wi-Fi networks [39], typically support 2 to 4 distinct transmission paths. To evaluate the robustness of OLMS in challenging environments, we first consider 4 paths between the two hosts, as suggested in the previous study [40]. To control the maximum number of supported paths, we configure the routing table such that each sender's Ethernet port can only communicate directly with only one receiving host's Ethernet port. This creates 4 disjoint paths: eth0a-Router-eth0b, eth1a-Router-eth1b, eth2a-Router-eth2b, and eth3a-Router-eth3b). Note that paths are not required to be disjoint, as overlapping paths are common in practical network scenarios. This configuration aligns with the setups used in the recent studies [36], [41].

2) Path Characteristic Control: We configure the Ethernet ports at the sender to vary the path characteristics using the network emulation function `netem`[5] and the traffic control tool `tc`. Specifically, for each of the four successfully established paths, we set different queuing disciplines using `tc` on each of the four Ethernet ports at the sender to create heterogeneity in the path characteristics, R_{rtt} and B_{bw}.

B. Application I: MaxRTT Constrained Multi-path Scheduling

To investigate the performance of the *maxRTT constrained* application, we initiate a large file (5 GB) transfer from the sender to the client in our network setup described in Section VI-A. Upon establishing an MPQUIC connection with four paths, we consider scheduling the optimal path for each packet from 4 paths ($K = 4$) using Algorithm 1 with components specified in *maxRTT constrained* multi-path scheduling in Section IV-A.

Then we use `tc` to configure the Ethernet ports at the sender to control the path characteristics of the 4 paths. Specifically, the delays of the four Ethernet ports at the sender are set to 20 ms, 30 ms, 50 ms, and 60 ms respectively, with an additional jitter of 5 ms introduced on each path to simulate realistic delay variations. The available bandwidth for these ports is limited at 30 Mbps, 20 Mbps, 60 Mbps, and 40 Mbps, respectively. The loss rates for these paths are set to 1.00%, 1.19%, 1.50% and 1.56%, respectively. We set the maximum RTT threshold for the *maxRTT constrained* application to 50 ms, i.e., on average, the maximum RTT of the scheduled path should not exceed 50 ms. To facilitate comparison, we normalize the four RTTs and the threshold to [0,1] using a scaling factor of 100 ms, and we normalize the four available bandwidths to [0,1] using a scaling factor of 100 Mbps. Thus, the R_{rtt} vector for the four paths is $\boldsymbol{r} = (0.2, 0.3, 0.5, 0.6)$, the B_{bw} vector is $\boldsymbol{b} = (0.3, 0.2, 0.6, 0.4)$, and the maximum RTT threshold is $C_r = 0.5$. For each packet, we select a path randomly from a set of 2 ($L = 2$) optimal paths sampled with the path scheduling vector $\boldsymbol{v}_t$. This random selection from a set of optimal paths ensures that packets are transmitted via high-quality paths in case the primary path becomes unavailable. By pre-selecting a subset of optimal paths, the scheduler reduces the risk of falling back to a suboptimal path selected by minRTT. By solving (5) for the optimal path scheduling vector $\boldsymbol{v}^*$ and the optimal utility $\mathsf{OPT}_{\max}$, we have $\boldsymbol{v}^* = (0.17, 0, 1.0, 0.83)$ and $\mathsf{OPT}_{\max} = 0.98$.

[5]https://man7.org/linux/man-pages/man8/tc-netem.8.html

We compare OLMS against other schedulers by examining the total throughput of selected paths and the RTT of the MPQUIC data connection. We estimate the RTT of the scheduled path by reading from the measurements of smoothed RTT. We estimate the throughput of the scheduled paths using the ratios of the size of the congestion window to the smoothed RTT. The throughput of the selected path quantifies the total throughput of the paths selected by the scheduler in each time slot. For simplicity of illustration, we present the first 20-second experiment results averaged over 50 times of the same 5 GB file transfers. These averaged results provide a concise overview of the performance of OLMS and other schedulers in managing bandwidth and latency in the MPQUIC data connection.

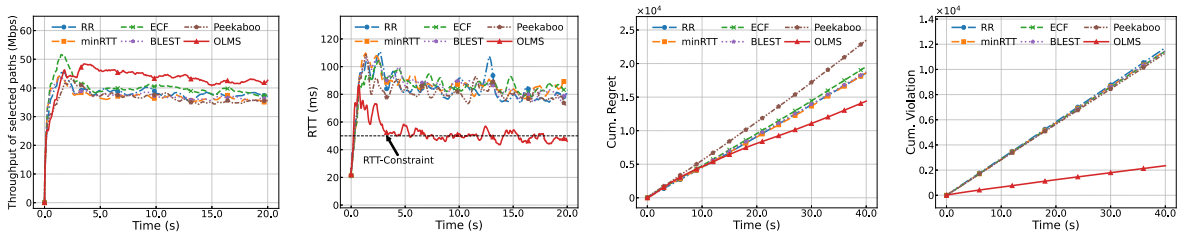
As shown in Fig. 3(a), OLMS achieves 8.42%–18.71% higher throughput compared to other schedulers. The average bandwidth of minRTT is lower than RR, which can be attributed to the heterogeneity of path characteristics. ECF also has a lower bandwidth given the heterogeneous path characteristics as it mainly focuses on minimizing the completion time of each packet. We also evaluate the bandwidth utilization of OLMS across four paths, which are 71%, 22%, 98%, and 77% respectively. It demonstrates that OLMS can effectively select the high-quality paths. Fig. 3(b) shows the measured RTTs of different schedulers in the MPQUIC data connection. The RTTs of our OLMS framework are all around the threshold 50ms ($C_r = 0.5$). In contrast, the RTTs of other schedulers (Peekaboo, ECF, minRTT, and BLEST) have very large fluctuations since they do not respect the maximum RTT constraint. RR experiences the highest latency with a 90 ms RTT in the *maxRTT constrained* multi-path scheduling application. Peekaboo, ECF, minRTT, and, BLEST also have higher latencies than OLMS. The reason is that they improperly schedule the heterogeneous paths and cause queue backlog thereby increasing the latency. OLMS outperforms other schedulers as it provides more appropriate scheduling than other schedulers with its *Constrained Multi-path Scheduling* component using accurate estimates of bandwidth and RTT from its *Path Characteristic Estimation* component.

To put things in perspective, we show the cumulative regret and cumulative violation of our OLMS framework and other schedulers in Fig. 3(c) and (d), respectively. Our OLMS framework has 22.50%–38.64% less cumulative regret and 79.52%–79.98% less cumulative violation than other schedulers.

Overall, the experiments show that our OLMS framework consistently schedules better paths compared to other schedulers by actively adjusting path scheduling during large file transfers. This scenario serves as a representative application of *maxRTT constrained* multi-path scheduling. The results demonstrate the effectiveness of OLMS in optimizing path scheduling and improving overall performance in real-world network environments.

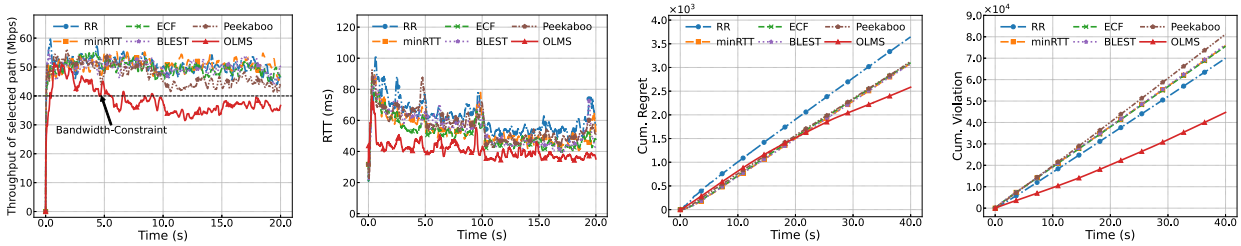
C. Application II: Bandwidth Constrained Multi-path Scheduling

To investigate the performance of *bandwidth constrained* application in OLMS, we initiate a 500 MB file transfer and run Algorithm 1 with the three components specified in



(a) Throughput (b) RTT (c) Cumulative regret (d) Cumulative violation

Fig. 3. Experiment results of *maxRTT constrained* multi-path scheduling with with $C_r = 0.5$ (Maximum RTT is constrained at 50ms).



(a) Throughput (b) RTT (c) Cumulative regret (d) Cumulative violation

Fig. 4. Experiment results of *bandwidth constrained* multi-path scheduling with $C_b = 0.4$ (Bandwidth throttling threshold is set to 40 Mbps). .

Application II using the network setup in Section VI-A. Similar to the experiment setup in Application I, we consider scheduling the optimal paths from 4 ($K = 4$) paths.

In the *bandwidth constrained* application, the available bandwidths of the four Ethernet ports are limited to 30 Mbps, 10 Mbps, 40 Mbps, and 60 Mbps, respectively. The bandwidth throttling threshold is set to 40 Mbps, ensuring that the throughput of the scheduled path does not exceed this threshold on average. The delays of the four Ethernet ports at the sender are configured to be 30 ms, 30 ms, 40 ms, and 10 ms, respectively. We normalize the available bandwidths and the bandwidth throttling threshold to the range $[0, 1]$, and similarly normalize the RTTs to the range $[0, 1]$. Thus, the B_{bw} vector for the four paths is $\boldsymbol{b} = (0.3, 0.1, 0.4, 0.6)$, the R_{rtt} vector is $\boldsymbol{r} = (0.3, 0.3, 0.4, 0.1)$, and the bandwidth throttling threshold is $C_b = 0.4$. We consider a scenario where a path is selected for each packet randomly from a set of 2 ($L = 2$) optimal paths sampled from the 4 candidate paths using $\boldsymbol{v}_t$. Solving (9) reveals the optimal scheduling vector $\boldsymbol{v}^* = (1, 1, 0, 0)$ and the optimal utility $\mathsf{OPT}_{\min} = 0.3$. This indicates that the optimal path scheduling algorithm would always schedule packets between the first two paths to ensure that the bandwidth is throttled at the bandwidth throttling threshold.

Fig. 4 shows the experiment results of the RTTs, the throughput of selected paths, the cumulative regret, and the cumulative violation averaged over 50 times of a 500 MB file transfer. In Fig. 4(a), OLMS initially violates the bandwidth throttling constraint C_b before $t = 5$ as it lacks information of the path characteristics. After $t = 5$, the throughput of selected paths of OLMS constantly adheres to the bandwidth throttling threshold C_b. In contrast, other schedulers fail to comply with the bandwidth throttling constraint and aggressively utilize the available bandwidth. Thus, their throughput of selected paths exceeds the threshold C_b. Similarly, the bandwidth utilization of OLMS across four path is 99%, 76%, 33%, and 25% respectively. It shows that OLMS can select high-quality paths while adhering to the bandwidth throttling threshold. As shown in Fig. 4(b), our OLMS framework achieves the lowest latency with RTTs around 40 ms among all schedulers. Other schedulers have much higher RTTs, primarily due to the introduction of queue delays at the paths resulting from inappropriate path scheduling decisions. This shows the effectiveness of OLMS in maintaining adherence to bandwidth throttling constraints over time.

Fig. 4(c) and (d) show the cumulative regret and cumulative violation of different schedulers. OLMS achieves the lowest regret, and demonstrates a 15.97%–29.12% reduction in cumulative regret and a 36.14%–44.98% reduction in cumulative violation compared to other schedulers.

D. Other Experiments

In this subsection, we investigate the effectiveness of OLMS in web browsing, and study the impact of flow size and network dynamics on our OLMS framework. At last, we investigate the effectiveness of OLMS when competing with other flows.

1) Web Browsing: In this experiment, we consider a typical mobile network scenario where each client can communicate with the server via both LTE/5G and Wi-Fi networks. Specifically, the Wi-Fi and LTE links have bandwidths of 20 Mbps and 5 Mbps, RTTs of 20 ms and 60 ms, and packet loss rates of 1.56% and 0.8%, respectively. We evaluate the performance of different schedulers in browsing the home pages of Google and YouTube, respectively. Specifically, we host copies of the home pages, google.com and youtube.com, on an MPQUIC-enabled web server by scraping down all the web objects from the websites.

TABLE I
WEBSITE INFORMATION FOR GOOGLE AND YOUTUBE

Website URL	# of objects	Page size (MB)
https://www.google.com	9	1.67
https://www.youtube.com	67	16.6

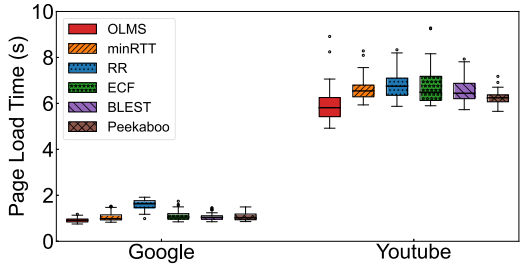


Fig. 5. Page load time of Google and YouTube.

The number of objects and the page size for the two websites are detailed in Table I. Then we deploy a web client to retrieve the web pages. To assess the effectiveness of OLMS framework in web browsing, we analyze the page load time (PLT), i.e., the completion time of all web objects. For brevity, we only present the experiment results of the *maxRTT constrained*multi-path scheduling application using the path characteristics setting in Application I. For each website, we repeat the web browsing 50 times and compare the PLT of OLMS with other schedulers. This comparative analysis provides insights into the effectiveness of OLMS in optimizing webpage loading times and enhancing the browsing experience for users.

As shown in the box plot in Fig. 5, when browsing the Google home page, OLMS achieves a slightly better PLT, reducing it by up to 8% compared to other schedulers. It is worth noting that the relatively small number of objects on the Google home page limits OLMS's ability to gather sufficient statistics for accurately learning the path characteristics. In contrast, for the home page of YouTube, which features a significantly larger number of objects, OLMS achieves the shortest PLT with 12.3%–25.9% reduction compared to other schedulers, as shown in Fig. 5. This significant improvement can be attributed to OLMS's ability to collect more statistics and provide more accurate estimates of path characteristics when dealing with a larger number of web objects. In summary, our findings in Fig. 5 show that OLMS improves web browsing speed for content-rich websites.

2) Impact of Flow Size: As reported in [42], [43], network traffic exhibits a high variability in terms of the flow sizes. To study the impact of flow size on OLMS, we use files of different sizes, including 5 MB, 200 MB, and 1500 MB, in the file transfers. For brevity, we only present the experiment results of the *maxRTT constrained* application using the path characteristics setting in Application I. For each file size, we repeat the file transfer process 50 times and compare the flow completion time of OLMS with that of other schedulers. This comparative study allows us to assess how OLMS performs under varying flow sizes and its ability to adapt to different network traffic.

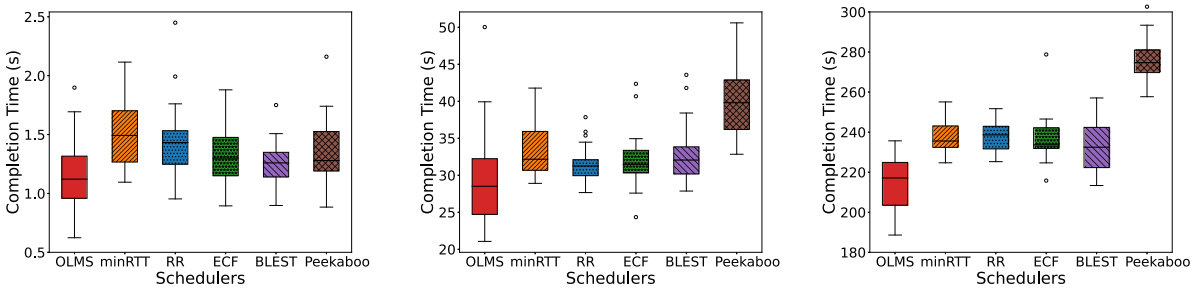
As shown in Fig. 6, across all flow sizes, OLMS consistently achieves the lowest flow completion time compared to other schedulers. Especially, it has a much lower flow completion time compared to minRTT for large flow sizes (1500 MB). The reason is that OLMS has more time to learn the path characteristics and schedule the optimal paths when the flow size is large. Another observation is that the advantage of Peekaboo in flow completion time diminishes as the flow size increases. It can be attributed to Peekaboo struggling to effectively learn the weights associated with path characteristics in scenarios. In summary, Fig. 6 shows that OLMS facilitates lower flow completion times and is effective across different flow sizes.

3) Impact of Network Dynamics: To study the impact of network dynamics on different schedulers, we conduct experiments under dynamic network conditions with varying bandwidth and RTT. Note that we have integrated the *Path Characteristic Measurements Monitor* component into OLMS to detect abrupt changes of path characteristics. We refer to the scheduler OLMS with *Path Characteristic Measurements Monitor* enabled as "OLMS with PM". Specifically, we set the parameters $q = 20$, $\delta = 3$ and $h = 3$ in Algorithm 2. For brevity, we only present the results of the *maxRTT constrained* multi-path scheduling application using the path characteristics setting in Application I. We repeat a 200 MB file transfer experiment 50 times and compare the average file transfer completion times of different schedulers under the following two dynamic network settings:

- *Dynamic Bandwidth.* Every 10 seconds, the bandwidth of both path 1 and path 2 drops to 1 Mbps in the first 5 seconds, then restores back to 30 Mbps and 20 Mbps in the subsequent 5 seconds. Meanwhile, the bandwidth of path 3 and path 4 remains at 60 Mbps and 40 Mbps for the first 5 seconds, then both drop to 2 Mbps in the subsequent 5 seconds.
- *Dynamic RTT*. The RTTs of path 1 and path 3 remain constant throughout the experiment. Every 10 seconds, the RTT of path 2 increases to 100 ms in the first 5 seconds, then returns to 30 ms in the subsequent 5 seconds. Similarly, the RTT of path 4 decreases to 10 ms in the first 5 seconds, then increases back to 60 ms in the following 5 seconds.

As shown in Fig. 7, OLMS with PM consistently achieves the shortest file transfer completion time compared to other schedulers in both dynamic bandwidth and dynamic RTT network environments. Conversely, OLMS without PM demonstrates poorer performance compared to other schedulers. The reason is that OLMS without PM heavily depends on historical information for estimating path characteristics, which results in slower responsiveness to dynamic network environments. These results suggest that the *Path Characteristic Measurements Monitor* component can effectively detect abrupt network changes, and intervene in the learning of path characteristics, thereby improving the scheduling of multiple paths in dynamic network environments. Overall, our findings in Fig. 7 indicates that OLMS with *Path Characteristic Measurements Monitor* component can adapt to dynamic network environments effectively.

4) Multi-Flow Scenarios: To evaluate the effectiveness of OLMS in multi-flow scenarios, we conduct experiments in a dumbbell topology, where two client-server pairs, (c_1, s_1) and



(a) Flow size: 5MB (b) Flow size: 200MB (c) Flow size: 1500MB

Fig. 6. Flow completion times of different schedulers under different flow sizes.

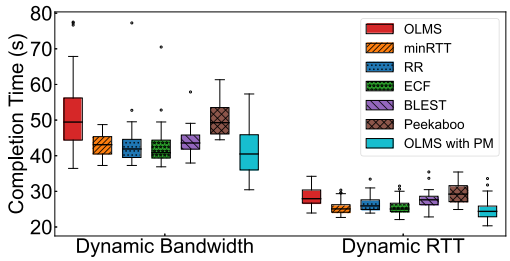


Fig. 7. File transfer completion time under different dynamic network settings.

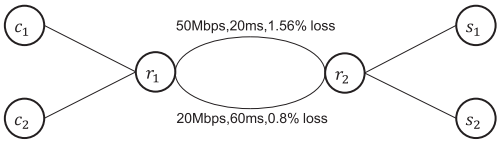


Fig. 8. The dumbbell topology with two paths.

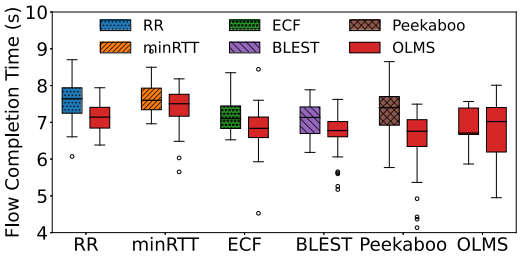


Fig. 9. Flow completion times of OLMS competing with other schedulers.

(c_2, s_2), each have two transmission paths. The path characteristics, including bandwidth, RTT, and loss rate, are shown in Fig. 8. For brevity, we present only the results for the *maxRTT constrained* application, with the maximum RTT threshold set to 50 ms. Each flow transmits a 10 MB file. The scheduler for the first flow (c_1, s_1) is varied across all available schedulers, while the second flow uses OLMS. Each file transfer experiment is repeated 50 times per flow, and we compare the flow completion times under different scheduler combinations.

As shown in Fig. 9, when both flows adopt OLMS, they achieve nearly identical flow completion times due to similar scheduling decisions. However, when the OLMS flow competes

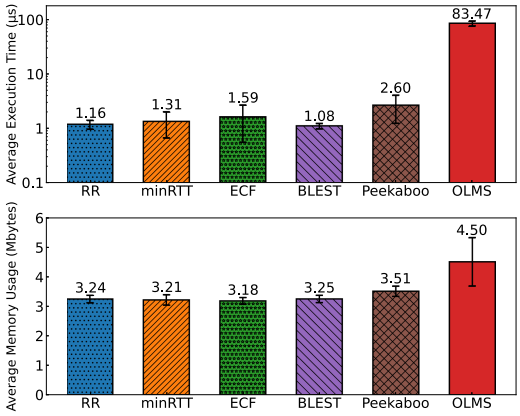


Fig. 10. Average execution time of selecting paths and average memory usage of different schedulers.

with the minRTT flow, the flow completion times for both flows increase compared to other scheduler combinations. This occurs because both OLMS and minRTT prioritize the Wi-Fi link, which has a lower RTT, leading to congestion on the Wi-Fi link. In summary, OLMS reduces flow completion times by 4.22%–10.26% compared to other schedulers. This improvement is attributed to OLMS's tendency to select the Wi-Fi link, which satisfies the maximum RTT constraint while offering higher bandwidth. These results indicate that OLMS achieves lower flow completion times when competing with other flows.

E. Discussions

1) Overhead in OLMS: In this subsection, we evaluate the overhead introduced by solving LP problems during path selection for each packet. The evaluation focuses on the delay caused by solving the LP problems and the memory usage of the MPQUIC connection. The experiments are conducted using the network topology shown in Fig. 8.

LP solving time: We evaluate the average execution time of the `selectPath()` function for each scheduler, with the result presented in Fig. 10. While OLMS requires more time to solve LP problems than other schedulers, this does not negatively impact overall data transmission performance, provided the LP solving speed exceeds the data transmission speed (see the discussion in Section VI-E3). In typical scenarios, the scale of

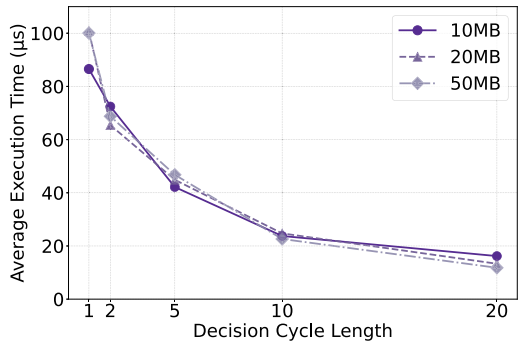


Fig. 11. Average execution time of selecting paths in OLMS under different decision cycle lengths.

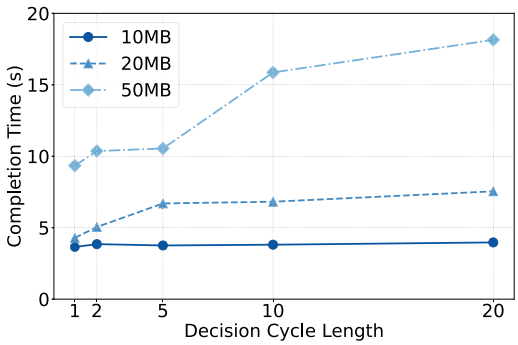


Fig. 12. Average completion time in OLMS under different decision cycle lengths.

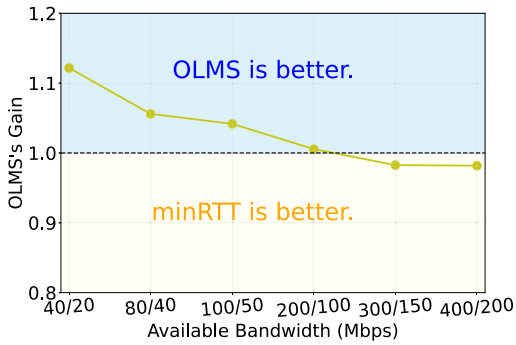


Fig. 13. Gain of OLMS under different available bandwidth.

LP problems is small, as only 2–4 paths are usually involved. Additionally, hardware acceleration techniques, such as those described in [44], [45], [46], can efficiently solve LP problems, thereby mitigating the impact of LP solving overhead on transmission performance. To further reduce delays associated with LP solving, we propose a periodic solving technique detailed in Section VI-E2.

Memory usage: We measure the memory usage of different schedulers implemented in Golang using the `menstats` tool. Fig. 10 shows that OLMS has slightly higher memory usage than other schedulers, mainly due to solving LP problems. However, this does not affect the overall performance of OLMS, as modern terminal devices and smart NICs generally have enough memory to accommodate these requirements.

2) Cyclic LP Solving: To mitigate the potential extra delay and memory usage caused by frequent LP solving in OLMS, we propose a *cyclic LP solving* mechanism to reduce the number of LP computations. Instead of solving an LP problem for every packet, this mechanism allows multiple decision-making (path selection) steps to share a single LP solution across several packets. Specifically, LP problems are solved cyclically for a fixed number of packets, referred to as the *decision cycle*, with the solution reused for all decisions within that cycle. For instance, a decision cycle of 5 applies the same LP solution to the decisions for five consecutive packets.

We evaluate the impact of different decision cycle lengths on the average execution time for path selection and the average completion time of OLMS using the network topology described in Section VI-E1. The results, averaged over 50 experiments with varying file sizes from 10 MB to 50 MB, are presented in Figures 11 and 12. The results demonstrate that the *Cyclic LP solving* mechanism effectively reduces the delays caused by LP solving. When the decision cycle length is less than five, the mechanism significantly reduces delays while maintaining performance comparable to the optimal solution. However, as the decision cycle length increases, performance may degrade due to the suboptimal reuse of LP solutions.

3) Impact of the Overhead: To evaluate the impact of overhead on OLMS's overall performance, we analyze the gain of OLMS, defined as the ratio of the completion time of minRTT to that of OLMS. A gain greater than 1 indicates that OLMS achieves a smaller completion time, whereas a gain less than 1 indicates better performance by minRTT. Using the network topology in Fig. 8, we conduct experiments under varying network conditions. The delay and loss rates for the two paths are consistent with those in Fig. 8. The available bandwidth is limited to 40 Mbps to 400 Mbps for one path and 20 Mbps to 200 Mbps for the other. The results are presented in Fig. 13. OLMS outperforms minRTT when bandwidth is low (below 200 Mbps). In these scenarios, the benefits of solving the LP problem outweigh the overhead introduced. However, as bandwidth exceeds 200 Mbps, the difference in completion time becomes negligible. At higher bandwidths, the delay caused by solving the LP problem begins to hinder OLMS's data transmission efficiency. Nevertheless, in current mobile network, available bandwidth is typically below 200 Mbps, making OLMS both practical and advantageous in real-world scenarios. Additionally, advancements in hardware acceleration are expected to reduce LP solving time, further mitigating the overhead and enhancing the performance of OLMS in high-bandwidth networks.

VII. Related Work

In the pioneering work [2] on MPTCP, Raiciu et al. propose the minRTT scheduler for path scheduling. Since then, minRTT has been the de facto path scheduler of MPTCP and MPQUIC as it helps to reduce delays for general applications. Recently, there have been many studies on the improvement of the path

scheduler of multi-path transport protocols from different aspects. Lim et al. [47] design the Earliest Completion First (ECF) scheduler to manage heterogeneous paths and improve the utilization of the fastest path. Ferlin et al. [48] propose the Blocking Estimation-based Scheduler (BLEST) which aims to avoid the head-of-line blocking by reducing the usage of slow path. Later on, Shi et al. [8] propose the Slide Together Multipath Scheduler (SMTS) that pre-allocates packets to send over the fast path to improve the throughput of MPTCP under heterogeneous networks. [49] shows that MPTCP does not handle asymmetric paths well and the data scheduler cannot provide low-latency transmission. Ferlin et al. [50] incorporates the forward error correction (FEC) into the MPTCP scheduler to support latency-sensitive applications under lossy links. Paasch et al. [10] conduct an extensive study on different schedulers and suggest that RTT-based path schedulers have limitations. Peng et al. [51], [52] propose a fluid model to analyze path scheduling in MPTCP and design an algorithm to balance among the TCP-friendliness, responsiveness, and window oscillation. Above studies on path schedulers of multi-path transport protocols are all conducted for applications without any constraints. In our work, we fill the void by designing the OLMS framework where users can specify different requirements for network applications.

Attracted by the effectiveness of learning algorithms, many works try to incorporate learning models to optimize the performance of network protocols. For example, PCC [22], PCC Vivace [19], PCC Proteus [20] apply online learning techniques to improve the performance of single-path TCP congestion control. Additionally, works like MPCC [53] and MPLibra [54] focus on online learning multi-path transport design for MPTCP. Some studies explore the use of deep neural networks [55] or reinforcement learning techniques (SmartCC [56]) to address challenges in MPTCP congestion control. Furthermore, there are works that employ online learning techniques in the design of multi-path scheduler, such as Peekaboo [36], OLAPS [57], and others leveraging contextual bandit approaches [58]. Although these works can achieve high performance using different learning algorithms, they neglect the user-defined requirements and have no theoretical guarantee, which are essential for network applications.

In the online learning literature, our OLMS framework is related to the multi-armed bandit (MAB) models first proposed in [59]. The classical MAB models [14], [15] mainly focus on minimizing the regret and finding the single optimal arm. Our OLMS framework differs from these models by considering the constraints in selecting the arms and introducing both the regret and violation metrics for performance measurement. OLMS provides theoretical guarantees on both sublinear regret and sublinear violation. In this context, OLMS is related to the algorithms introduced in [24], where the authors address long-term constraint satisfaction and propose UCB-based algorithms capable of cancelling violations during the learning process. However, OLMS adopts a stricter violation definition where the constraint violations are considered in each time slot without any cancellations. Besides, OLMS is developed from Thompson Sampling [18] and offers a more versatile approach without relying on any prior distributions.

VIII. Conclusion and Future Work

To our best knowledge, we are the first to present an online learning multi-path scheduling (OLMS) framework with user-defined requirements for multi-path transport protocols. In our OLMS framework, we have designed a general online learning algorithm capable of scheduling paths to meet different requirements in the maxRTT constrained, and bandwidth constrained multi-path scheduling applications. We have proved that OLMS achieves both sublinear regret and sublinear violation for the two applications. Through extensive experimentation conducted on a prototype system built upon MPQUIC, we have demonstrated the effectiveness of our OLMS framework in multi-path scheduling for various applications while satisfying different requirements. Moreover, our experiments have highlighted OLMS's efficacy in web browsing, its adaptability to different flow sizes, and its robust performance in dynamic network environments, without incurring large overhead.

Our future work should continue to improve the design of our OLMS framework. Specifically, our research should integrate additional path information, such as buffer size and out-of-order degree, into OLMS for more refined path scheduling strategies, and deploy OLMS on smart NICs to further optimize performance.

References

[1] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *Proc. IEEE*, vol. 98, no. 1, pp. 100–122, Jan. 2010.

[2] C. Raiciu et al., "How hard can it be? designing and implementing a deployable multipath TCP," in *Proc. 9th USENIX Symp. Netw. Syst. Des. Implementation* 2012, pp. 399–412.

[3] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.* (CoNEXT), 2017, pp. 160–166.

[4] Y. Lu et al., "Multi-path transport for RDMA in datacenters," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 357–351.

[5] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM 2011 Conf.*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 266–277.

[6] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[7] J. Han et al., "FMPTCP: Achieving high bandwidth utilization and low latency in data center networks," *IEEE Trans. Commun.*, vol. 72, no. 1, pp. 317–333, Jan. 2024.

[8] H. Shi et al., "STMS: Improving MPTCP throughput under heterogeneous networks," in *Proc. Annu. Tech. Conf.*, 2018, pp. 719–730.

[9] S. Barré, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *Proc. Int. Conf. Res. Netw.*, Springer, 2011, pp. 444–457.

[10] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop*. ACM, 2014, pp. 27–32.

[11] B. Y. Kimura, D. C. Lima, and A. A. Loureiro, "Packet scheduling in multipath TCP: Fundamentals, lessons, and opportunities," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1445–1457, Jan. 2020.

[12] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[13] C. Wang, J. Guan, T. Feng, N. Zhang, and T. Cao, "BitLat: Bitrate-adaptivity and latency-awareness algorithm for live video streaming," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 2642–2646.

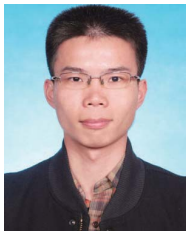
[14] P. Auer, I. for Theoretical, and C. Science, *The Non-Stochastic Multi-Armed Bandit Problem*. Philadelphia, PA, USA: SIAM, 1995.

[15] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, 2002.

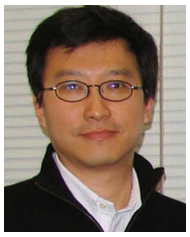
[16] R. Mittal et al., "Timely: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, ACM, 2015, vol. 45, no. 4, pp. 537–550.
[17] S. Arslan, Y. Li, G. Kumar, and N. Dukkipati, "Bolt:{Sub-RTT } congestion control for { Ultra-Low} latency," in *Proc. 20th USENIX Symp. Networked Syst. Des. Implementation* (NSDI 23), 2023, pp. 219–236.
[18] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.
[19] M. Dong et al., "PCC Vivace: Online-learning congestion control," in *Proc. USENIX Symp. Networked Syst. Des. implementation*, 2018, pp. 343–356.
[20] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "PCC proteus: Scavenger transport and beyond," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2020, pp. 615–631.
[21] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.
[22] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. USENIX Symp. Networked Syst. Des. implementation*, 2015, pp. 395–408.
[23] R. Jenatton, J. C. Huang, and C. Archambeau, "Adaptive algorithms for online convex optimization with long-term constraints," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 402–411.
[24] K. Cai, X. Liu, Y.-Z. J. Chen, and J. C. Lui, "Learning with guarantee via constrained multi-armed bandit: Theory and network applications," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5346–5358, Sep. 2023.
[25] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *J. ACM (JACM)*, vol. 53, no. 3, pp. 324–360, 2006.
[26] "Is multi-path transport suitable for latency sensitive traffic," *Comput. Netw.*, vol. 105, no. C., pp. 1–21, Aug. 2016.
[27] L. Li et al., "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proc. SIGCOMM*, 2018, pp. 161–175.
[28] "Apple opens multipath TCP in iOS11," Accessed: Nov. 1, 2021. [Online]. Available: https://www.tessares.net/highlights-from-advances-in-networking-part-1/
[29] " In Korean, *multipath TCP is pronounced GIGA path*," Accessed: Nov. 1, 2021. [Online]. Available: http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html
[30] Y. Sui, V. Zhuang, J. W. Burdick, and Y. Yue, "Multi-dueling bandits with dependent arms," *arXiv:1705.00253*, 2017.
[31] H. Karloff, *Linear Programming*. Berlin, Germany: Springer Science & Business Media, 2008.
[32] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. On Netw.*, vol. 21, no. 5, pp. 1651–1665, May 2013.
[33] C. Raiciu et al., "How hard can it be? designing and implementing a deployable multipath {TCP}," in *Proc. 9th USENIX Symp. Networked Syst. Des. implementation* (NSDI 12), 2012, pp. 399–412.
[34] Y. sup Lim, "ECF: An MPTCP path scheduler to manage heterogeneous paths (poster)," in *Proc. 13th Int. Conf. Emerg. Netw. Experiments Tech.*, 2017, pp. 147–159.
[35] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 302–315, Jan. 2018.
[36] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2295–2310, Oct. 2020.
[37] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proc. 8th Int. Conf. Emerg. Netw. experiments Technol.*, 2012, pp. 253–264.
[38] I. Qualcomm Technologies, "Unleashing full potential for simultaneous 5 G cellular connections: Qualcomm DSDA GEN 2 with dual data," 2023, Accessed: Nov. 20, 2024. [Online]. Available: https://www.qualcomm.com/news/onq/2023/05/unleashing-full-potential-for-simultaneous-5g-cellular-connections-qualcomm-dsda-gen-2-with-dual-data
[39] I. Qualcomm Technologies, "Qualcomm fastconnect6900 system product brief," Qualcomm Technologies, Inc., 2020, Accessed: Nov. 20, 2024. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/fastconnect-6900-product-brief.pdf
[40] B. Hesmans, H. T. Viet, R. Sadre, and O. Bonaventure, "A first look at multipath-TCP.org : Subflows," Accessed: Nov. 20, 2024. [Online]. Available: http://blog.multipath-tcp.org/blog/html/2014/12/17/measurements3.html
[41] C. Song, B. Han, R. Li, X. Han, C. Liu, and J. Su, "Aquilas: Adaptive QOS-oriented multipath packet scheduler with hierarchical intelligence for QUIC," in *Proc. IEEE 44th Int. Conf. Distrib. Comput. Syst.*, 2024, pp. 485–495.
[42] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.
[43] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 123–137.
[44] H. Lu et al., "CuPDLP-C: A strengthened implementation of cuPDLP for linear programming by C language," 2023, *arXiv:2312.14832*.
[45] N. Corporation, "NVIDIA cuOpt user guide: Introduction," 2024, Accessed: Dec. 09, 2024. [Online]. Available: https://docs.nvidia.com/cuopt/user-guide/introduction.html
[46] H. Lu and J. Yang, "cuPDLP. jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia," 2023, *arXiv:2311.12180*.
[47] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. Experiments Technol.*, 2017, pp. 147–159.
[48] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "Blest: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Netw. Conf. Workshops*, May 2016, pp. 431–439.
[49] P. Hurtig, K. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 302–315, Jan. 2019.
[50] S. Ferlin, S. Kucera, H. Claussen, and O. Alay, "MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2005–2018, Oct. 2018.
[51] Q. Peng, A. Walid, and S. H. Low, "Multipath TCP algorithms: Theory and design," in *Proc. ACM SIGMETRICS Performance Evaluation Rev.*, vol. 41, no. 1. ACM, 2013, pp. 305–316.
[52] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Jan. 2016.
[53] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira, "MPCC: Online learning multipath transport," in *Proc. ACM CoNEXT*, 2020, pp. 121–125.
[54] H. Yu, J. Zheng, Z. Du, and G. Chen, "Mplibra: Complementing the benefits of classic and learning-based multipath congestion control," in *Proc. IEEE 29th Int. Conf. Netw. Protoc.*, 2021, pp. 1–11.
[55] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1325–1336, Jun. 2019.
[56] W. Li, H. Zhang, S. Gao, C. Xue, X. Wang, and S. Lu, "SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2621–2633, Nov. 2019.
[57] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, and D. S. Wei, "An online learning assisted packet scheduler for MPTCP in mobile networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 5, pp. 2297–2312, Oct. 2023.
[58] J. Zhao and J. Pan, "QoE-driven joint decision-making for multipath adaptive video streaming," in *Proc. IEEE 42nd Glob. Commun. Conf.*, 2023, pp. 128–133.
[59] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.

Kechao Cai (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2010 and 2013, respectively, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2019. He is currently an Assistant Professor with the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen, China. His research interests includes network protocol design and online learning algorithms.

Zhuoyue Chen received the B.S. degree in communication engineering in 2022 from Sun Yat-sen University, Shenzhen, China, where he is currently working toward the doctor's degree in information and communication engineering. His research interests include online learning algorithms and network protocol designs.



Jinbei Zhang (Member, IEEE) received the B.S. degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. From 2016 to 2018, he was a Postdoc with The Chinese University of Hong Kong, Hong Kong. Since 2018, he has been an Associate Professor with Sun Yat-sen University, Shenzen, China. His research interests include network virtualization and caching.



John C. S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), Los Angeles, CA, USA. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests include communication networks, system security (e.g., cloud security, mobile security, etc.), network economics, network sciences, large-scale distributed systems, and performance evaluation theory. He is also on the Editorial Board of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Journal of Performance Evaluation*, and *International Journal of Network Security*. From 2005 to 2011, he was the Chairman with CSE Department. He was the recipient of the various Departmental Teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He is also the Corecipient of the IFIP WG 7.3 Performance 2005 and IEEEIFIP NOMS 2006 Best Student Paper awards. He is an Elected Member of the IFIP WG 7.3, Fellow of the ACM, and Croucher Senior Research Fellow.