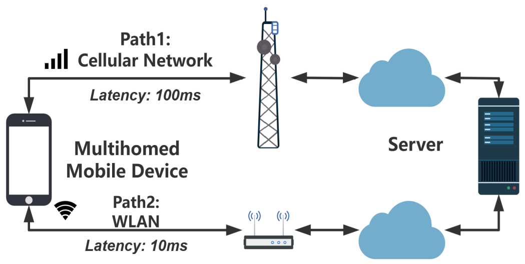
Optimizing multipath QUIC transmission over heterogeneous paths

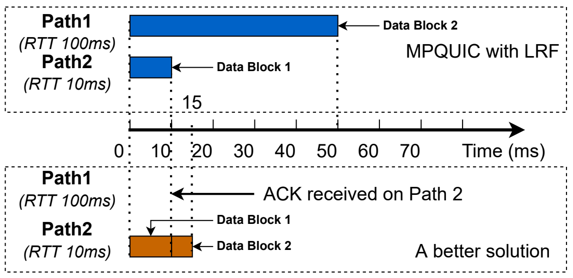
Graphical Abstract

Optimizing Multipath QUIC Transmission over Heterogeneous Paths

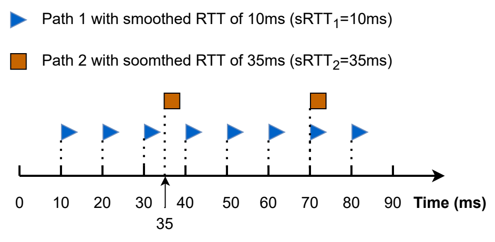
Hongxin Zeng, Lin Cui, Fung Po Tso, Zhen Zhang

Optimizing Multipath QUIC Transmission over Heterogeneous Paths





Mobile devices with multiple interfaces are prevailing yet current MPQUIC implementation's LRF scheduler faces performance degradation over heterogeneous paths


Path 1 with smoothed RTT of 10ms (sRTT1=10ms)
Path 2 with soomthed RTT of 35ms (sRTT2=35ms)
0
10
20
30
40
50
60
70
80
90
Time (ms)
35

Estimated Transfer Completion time (ETC) scheduler that simulates congestion window changes to calculate data partition for each path shortens the file completion time by up to about 29.6% according to results of Mininet based experiments.

Highlights

Optimizing Multipath QUIC Transmission over Heterogeneous Paths

Hongxin Zeng, Lin Cui, Fung Po Tso, Zhen Zhang

- We investigate the issue of performance degradation over heterogeneous paths for the LRF scheduler of MPQUIC.
- We formulate the problem model that assigns data partitions to heterogeneous paths and design an ETC scheduler for MPQUIC.
- We have implemented ETC based on *quic-go* and our extensive experiments show that ETC can shorten the file completion time by up to about 29.6% as compared to existing MPQUIC implementation.

Optimizing Multipath QUIC Transmission over Heterogeneous Paths

Hongxin Zeng[a], Lin Cui[a], Fung Po Tso[b], Zhen Zhang[a]

[a]*Department of Computer Science,Jinan University, Guangzhou, Guangdong, China*
[b]*Department of Computer Science, Loughborough, Leicestershire, UK*

Abstract

As a novel UDP-based transport protocol which supports stream multiplexing, QUIC is faster, more lightweight and flexible than TCP. With the prevalence of multi-homed devices such as smartphones with both WiFi and 4G/5G cellular connectivity, Multipath QUIC (MPQUIC) can effectively utilize multiple network interfaces (i.e., multiple paths) to improve transmission efficiency. Current MPQUIC implementation adopts the Lowest-RTT-First (LRF) scheduler which always selects the path with the lowest smoothed RTT among all available paths. However, we show that in networks with heterogeneous paths where network characteristics (e.g., RTT, loss rate) differ considerably, such scheduling scheme leads to unnecessary waiting on fast paths and bufferbloat, degrading overall transmission performance significantly. To use heterogeneous paths efficiently (i.e., to reduce the overall file transfer completion time), this paper proposes a novel scheduling mechanism that assigns data to paths with transfer simulation without causing much additional overhead. Extensive experiment results in Mininet demonstrate that the proposed scheduling mechanism can reduce the transfer completion time by up to 29.6% as compared to existing MPQUIC implementation.

Keywords:
Transport Protocol, Multipath QUIC, Scheduling

1. Introduction

QUIC is an emerging multiplexed transport protocol that runs on top of UDP, carrying more and more Internet traffic due to its improvements on both security and efficiency [1]. In the meantime, more and more communication devices are equipped with multiple network interfaces, e.g., smartphones with both WiFi and 4G/5G [2] interfaces. To exploit resources over multiple network interfaces, Multipath QUIC (MPQUIC) [3], as a multipath extension to QUIC, is proposed to enable concurrent multipath transfer over multiple network paths. With the flexibility inherited from QUIC, it is expected to have great potential to be widely deployed. For example, Alibaba has launched a large-scale A/B test of MPQUIC video transport in their production environments [4], demonstrating the feasibility and deployability of MPQUIC. Besides, because HTTP/3 uses QUIC as its underlying transport protocol [5], MPQUIC offers a great potential for the future web.

However, when the characteristics (e.g., loss rate or latency) of underlying paths differ, a multi-homed device encounters heterogeneous paths. One typical example is a smartphone with both WiFi and cellular networks (See Figure 1 for example). Since WiFi and cellular networks are designed for different scenarios in the first place, their network characteristics can be significantly different. For example, previous studies show that 90th percentile of LTE's latency is around 40*ms* [6], while the 90th percentile of WiFi's latency is generally around 20*ms* [7]. Thanks to the rapidly increasing number of mobile subscribers [8], heterogeneous paths will be more prevalent in the future, making it very important and inevitable for multipath transport protocols.

Current MPQUIC implementation uses the Lowest-RTT-First (LRF) scheduler [3], which treats paths whose congestion windows (CWNDs) are full as unavailable, precluding them from being scheduled for transmission. When making decisions, LRF prioritizes the available path with the lowest smoothed RTT for data transmission. However, always selecting the available path with the lowest RTT may lead to performance degradation when faced with heterogeneous paths.

This is because LRF only considers the RTT of available paths when making decisions and ignores the possibility that the overall transmission time can be shortened by waiting for the busy but fast paths to become available again. If a faster path, which is about to complete its data transmission, is deemed unavailable and excluded by LRF, a slower path will be selected for data transmission. Such a scheduling decision based on transient path availability over heterogeneous paths will significantly degrade the transfer performance of MPQUIC. In addition, only favoring paths with lower RTTs increases the chances of bufferbloat in the long run. Moreover, passively responding to packet loss (i.e., congestion window change due to packet loss) also contributes to LRF's performance degradation under heterogeneous paths. More detailed analysis and a motivating example on these issues are provided in Section 2.2.

In this paper, we propose *Estimated Transfer Completion time* (ETC) scheduler – a novel scheduling scheme which optimizes the overall file transfer completion time over hetero-

*Corresponding author
Email address: `tcuilin@jnu.edu.cn` (Lin Cui)

geneous paths. As inspired by Shortest Transfer Time First (STTF) [9], the main idea of ETC is to assign an appropriate amount of data to paths according to the data assignment calculated in the simulated transfer. ETC iterates through time points at which ACK packets are received on each path to simulate the transfer. Each time point indicates that data of a CWND's size has been successfully delivered in the last RTT on a specific path. And the path's CWND size should be updated at each time point. The iteration ends until the amount of delivered data in simulation is no smaller than a given amount. The amount of delivered data over each path in simulation can serve as the assignment that assigns data to paths in real transmission later. More details of ETC are described in Section 3.3.

In summary, the contributions of this paper are as follows:

- We investigate the issue of performance degradation over heterogeneous paths for the LRF scheduler of MPQUIC.
- We formulate the problem model that assigns data partitions to heterogeneous paths and design an ETC scheduler for MPQUIC.
- We have implemented ETC based on *quic-go* and our extensive experiments show that ETC can shorten the file completion time by up to about 29.6% as compared to existing MPQUIC implementation.

The remainder of this paper is organized as follows. Section 2 gives some necessary background on MPQUIC and a motivating example which shows the need for an improved scheduling mechanism for MPQUIC for heterogeneous paths. Section 3 elaborates the formulation of the scheduling problem and the design of the scheduling algorithm. Section 4 presents the implementation of ETC Section 5 describes our experiment environments and discusses the results. Related works are given in Section 6. Finally, Section 7 concludes the paper.

2. Background and Motivation

This section describes the basics of MPQUIC and analyses on heterogeneous paths.

2.1. MPQUIC

2.1.1. Overview

QUIC is a UDP-based transport protocol initially developed by Google. It is a multiplexed and secure general-purpose transport protocol that provides stream multiplexing, low-latency connection establishment and secured header and payload [10].

De Coninck et al. first presented the design and implementation of Multipath QUIC (MPQUIC) [3, 11], which is an extension of QUIC. Similar to Multipath TCP (MPTCP) [12], MPQUIC aggregates resources on multiple network interfaces. But as an extension to QUIC, MPQUIC is less sensitive to middlebox tampering compared to MPTCP, which leads to fewer compromises and simpler design. Therefore, MPQUIC is more likely to be widely deployed than MPTCP by nature.

2.1.2. Basic Designs of MPQUIC

The main design of MPQUIC is as follows:

Path Identification. Besides the four-tuple (i.e., source IP, source port, destination IP and destination port), MPQUIC defines a PathID field in the header, identifying the path on which the packet was sent.

Reliable Transmission. STREAM frame in QUIC contains an identification field and its offset, so the receiver can reorder received data based on StreamID and offset information easily. Each path has its own packet number space for ACK frames.

Path Management. Like QUIC, MPQUIC establishes a connection with a secure handshake over the initial path.

Packet Scheduling. Current MPQUIC implementation adopts the LRF scheduler which is the default scheduler of MPTCP in the Linux implementation [3]. A major difference is that, upon the completion of a connection establishment, the MPQUIC's LRF scheduler will duplicate packets and send them on all available paths instead of waiting until RTTs of all paths are probed.

2.2. MPQUIC over Heterogeneous Paths

The main goal of LRF scheduler is to prioritize the path with the lowest smoothed RTT to increase the throughput. Among all paths that meet the congestion window requirement, LRF scheduler will always select the lowest-RTT path for data transmission during each scheduling process. Since current MPQUIC implementation's scheduler only considers the available path with the lowest RTT, ignoring other faster paths that are about to become available, *it is likely to leave faster paths idle and increase the overall file transfer completion time.* This decision of scheduling data based on ephemeral network conditions can degrade transmission performance over heterogeneous networks. To demonstrate this issue clearly, a simple example is given as follows.

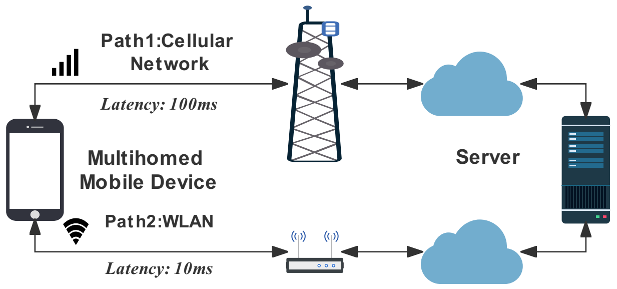


Figure 1: A common example of heterogeneous paths

As shown in Figure 1, it is common for today's mobile devices to have multiple network interfaces, such as cellular network and WiFi. When MPQUIC utilizes both of them for data transmission, it is faced with heterogeneous paths since WiFi and cellular connections have different latency and bandwidth. However, the LRF scheduler of MPQUIC cannot fully utilize the bandwidth of paths due to such heterogeneity.

Suppose there are two paths with the CWND size of W. Two data blocks, say data block 1 and data block 2, whose size are both W, arrive at the transmission buffer sequentially. Figure 2 depicts the detailed scheduling process of MPQUIC with LRF and an improved solution:

MPQUIC with LRF: Path2 will be selected to transfer data block 1 first (since it has lower RTT) and become unavailable due to its full congestion window. Then the scheduler will select path1 to transfer data block 2 as it is the available path with the lowest RTT at this point of time. As a result, the completion time will be around $50ms$ (half of path1's RTT).

An improved solution: If both blocks are transmitted over path2, the completion time will only be about $15ms$ (one and a half of path2's RTT).

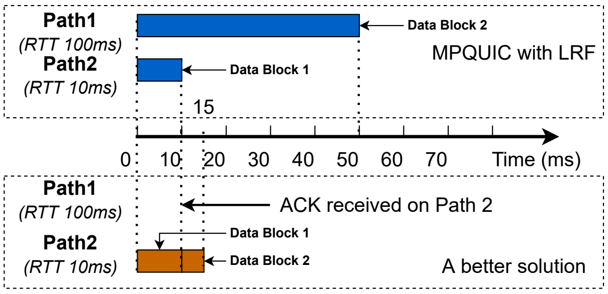


Figure 2: Transfer completion time is significantly prolonged under LRF

In addition, LRF is not responsive enough for packet loss on path. It only indirectly reacts to packet loss via the change of congestion window after loss. Assume that the loss rate of the lowest RTT path is high. Even though the CWND of the lowest RTT path is limited and selecting paths with high latency can lead to lower completion time, the LRF scheduler still favors it as a prior path to be selected.

Therefore, MPQUIC needs a new scheduler that can fully utilize heterogeneous paths with different RTTs and congestion states to solve the above problems.

3. ETC Design and Analysis

To optimize transmission performance of MQUIC over heterogeneous paths, the scheduling problem is first formulated and analyzed. Then, an efficient scheduling algorithm ETC is proposed to reduce the overall file transfer completion time.

3.1. Overview of MPQUIC Architecture

As an extension of QUIC, MPQUIC resides at transport layer and works in user space. As depicted in Figure 3, fundamental components of MPQUIC includes **Coupled Congestion Control**, **Path Manager** and **Packet Scheduler**.

Coupled Congestion Control is responsible for adjusting congestion windows of all paths based on session level information. It takes congestion windows of all paths into consideration and uses this information for congestion control of each path. In current MPQUIC implementation, the coupled congestion control is packaged as Opportunistic Linked Increases Algorithm (OLIA) Manager that implements the OLIA congestion control algorithm [13]. **Path Manager** is designed for path establishment, path migration, path information maintenance, etc. **Packet Scheduler** decides how to schedule data in buffer over underlying paths.

Thus, to cope with the performance degradation problem that happens over heterogeneous paths, a proper **Packet Scheduler** should consider path and congestion information.

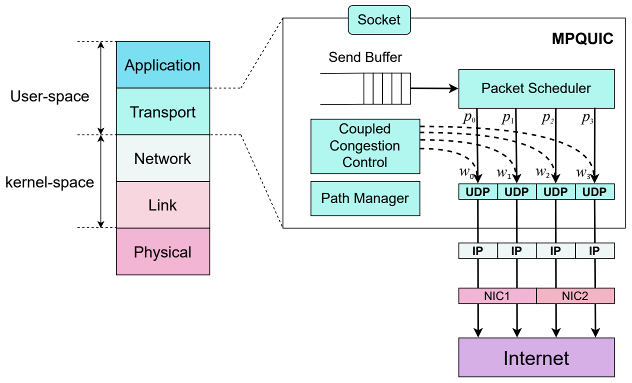


Figure 3: Overview of the MPQUIC architecture

Many schedulers have been proposed for multipath TCP and can be exploited by MPQUIC, as shown in Table 1. The scheduler of current MPQUIC, i.e., LRF, cannot handle heterogeneous paths well. Although other schedulers designed for MPTCP (e.g., DAPS [14], OTIAS [15] and ECF [16]) considers heterogeneous paths, they ignore congestion state information. Another state-of-art scheduler, BLEST [17], works well in preventing faster path from being idle to boost the performance over heterogeneous paths. It first selects a path using LRF. If the selected path is not the fastest path, it then evaluates whether transferring current segment on the selected path can be finish before the fastest path becomes available again. If so, this path will be selected for transfer eventually. Otherwise, it waits for the fastest path to become available again. As for STTF [9], it selects the best path fits for each segment. But in MPQUIC, the segment size is fairly small and cannot trigger the congestion state change of paths. More detailed description and analysis on STTF are provided in Section 6.

3.2. Modeling the Scheduling Problem

In this section, we provide a formal model for the problem of **Packet Scheduler** over heterogeneous paths.

Most schedulers select a path based on current network metrics whenever there are data to be sent in buffer. That is to say their decisions are made upon transient parameters for usually a small amount of data. This strategy results in the performance deterioration facing heterogeneity more or less without a global overview. To use heterogeneous paths efficiently, an ideal scheduler should assign a proper amount of data to each available path to make sure that *the longest transfer time among all selected paths is minimized.* To this end, it can instruct data scheduling over paths with previously computed data assignment to reduce the transfer completion time.

Assuming the amount of data to be sent is D and there are n active paths in the path set $\mathbb{P} = \{P_1, P_2, \ldots, P_n\}$. Among all active paths, there are $n-m$ ($m \leq n$) of them that are not suitable

Table 1: Overview Of Several Schedulers

Scheduler	**Input Metric**	**Approach**	**Complexity**	**Heterogeneity Optimization**	**Congestion State Consideration**
LRF	RTT	Select available path with the lowest RTT	Low	×	×
DAPS [14]	CWND	Schedules packets based on their sequence numbers for in-order receiving	Low	✓	×
OTIAS [15]	CWND, RTT	Estimates transfer time over all available paths and select one of the earliest path	Low	✓	×
ECF [16]	CWND, RTT and buffered data to be sent	Identifies whether a scheduling decision will make faster path idle	Low	✓	×
BLEST [17]	CWND, RTT and in-flight bytes	Estimates whether a scheduling decision will cause HoL-blocking and dynamically adapts scheduling to avoid blocking	Low	✓	×
STTF [9]	CWND, RTT and in-flight bytes	Calculate the expected transfer time of a segment considering the path characteristics of all available paths	High	✓	✓
ETC	CWND, RTT and in-flight bytes	Assign data over paths based on estimated transfer completion time	Low	✓	✓

to be selected for transmission, e.g., paths with significantly high latency or loss rate. D needs to be divided into m partitions and forms a data partition set $\mathbb{D} = \{D_1, D_2, \ldots, D_m\}$.

The scheduler should assign $\mathbb{D}$ to m paths. For an assignment A, let $A(D_i) = P_j$, if D_i is assigned to path P_j. For each $D_i \in \mathbb{D}$, it must be assigned to exactly one path in $\mathbb{P}$, i.e., $|A(D_i)|$=1. If $P_j = A(D_i)$, the transfer completion time T_i^j indicates the time that the ith data partition D_i required to be transferred on the jth path P_j.

Then, the objective of the scheduling problem is to find an appropriate partition $\mathbb{D}$ and the corresponding assignment A, which minimize the maximum transfer completion time of all partitions:

$$\begin{aligned} &\min \max T_i^j \\ &s.t. \begin{cases} |A(D_i)| = 1, \forall D_i \in \mathbb{D} \\ P_j = A(D_i) \wedge P_j \in \mathbb{P}, \forall D_i \in \mathbb{D} \end{cases} \end{aligned} \tag{1}$$

The first constraint indicates that each partition is assigned to exactly one path, and the second constraint ensures that the assigned path is an active path.

The optimal solution is to find a way of partitioning D based on some path information (e.g., RTT, CWND and congestion state) that the largest T_i^j can be minimized (as denoted in Equation(1)). This problem is a variation of the parallel task scheduling problem, which is an NP-hard problem [18]. Besides, it is not practical to obtain the total amount of data to be transferred because data is transferred in a streaming way and can be read into buffer in multiple times. Thus, it is infeasible to solve this problem in network stack.

3.3. *ETC Scheduler*

The design of the ETC scheduler incorporates the idea of multi-stated congestion control from the STTF scheduler. The main idea of ETC is to assign different amounts of data to different paths using transfer simulation on each path.

Before calculating the data assignment, all parameters of the CCA, for example, the size of congestion windows, should be preserved. The calculation involves congestion control related information and the RTT of each path. If these information for a certain path is not yet probed or unavailable, the path is skipped. Notably, ETC also uses the duplicating mechanism upon connection initialization in MPQUIC.

Each time point at which ACK packets are received on each path indicates that a CWND size of data has been successfully delivered during the last RTT. Then, the CWND size should be updated according to the adopted CCA. When updating CWND sizes, the congestion state (whether the path is in slow start or congestion avoidance) of the congestion control algorithm (CCA) for underlying paths is also taken into consideration to achieve more accurate estimation of completion time. With these time points, ETC can simulate the data transfer process.

Table 2: Notations and Symbols

Symbol	Meaning
D	Data to be sent
D_i	Divide data to m partitions, the ith partition
$\mathbb{D}$	The set of partitions
$\mathbb{P}$	Set of all available paths
P_j	The jth path in $\mathbb{P}$
A	An assignment that maps data partitions to corresponding paths
T_i^j	The time that D_i need to be transferred on P_j $(0 \le i < m, 0 \le j < n)$
W_s	The congestion window of path P_s
α_i	The factor defined in OLIA that guarantees responsiveness and nonflappiness
$t_{ess,i}$	Time for path P_i to exit slow start
$W_{begin,i}$	The congestion window of path P_i when ETC begins transfer time estimation
$SSTresh$	The slow start threshold
W_{init}	The initial window

ETC iterates through those time points chronologically. It collects the CWND size at each time point and updates the CWND size in a per-path manner. With all collected CWND sizes before update for one path, ETC can simulate data transfer on this path and calculate the amount of data sent in simulation. Once the total amount of data sent on all paths in simulation equals the amount of data to be sent, the iteration as well as the simulation end. After the simulation, the amount of data sent over paths during simulation serves as the assignment to instruct paths in the actual transfer process.

As mentioned in Section 3.2, the streaming way of transferring data makes it infeasible to get the total size of all data to be sent, so a basic scheduling unit S is introduced. It is a predefined amount of data used in each calculation as the sum of data. The scheduler will remove previous assignment and reschedule unsent data when the amount of data sent since the last computation reaches the basic scheduling unit or a signal of network congestion is detected. With this rescheduling mechanism, the scheduler proactively responds to path dynamics so that the interference from path status change caused by external events can be mitigated.

Detailed operations of the ETC scheduler are described in Algorithm 1. t_i marks the time point at which data are ACKed on path P_i in simulation. t_{cur} marks the current time point. The $GetSmoothedRTT(P_i)$ function gets the smoothed RTT for path P_i from MPQUIC's Path Manager. Line 5 initializes t_{cur} to infinity. Line 8~10 will bypass both initial path and paths that are not yet probed. Line 11~14 selects the path which is going to get ACKed at the closet next time point in simulation and updates t_{cur}. Line 16~17 increases the amount of data that should be sent on P_s by the CWND of P_s. After that, the CWND of P_s is changed accordingly with its CCA in Line 18. Line 19 updates the time point for P_s.

Figure 4 illustrates an example of the transfer simulation process without interruption of external events. Assume there

Algorithm 1 ETC Scheduling Algorithm

```
Input: Set of all paths $\mathbb{P}$; Basic scheduling unit $S$; Set of sizes all paths' congestion windows $W$;
Output: Table $M$ that maps a path to the amount of data assigned to it
1:  for $i = 0; i < n; i++$ do
2:      Preserve congestion control parameters of each path
3:  end for
4:  $t_{cur} \leftarrow \infty$
5:  while $S > 0$ do
6:      for $i = 0; i < n; i++$ do
7:          if $P_i$ is not probed or $P_i$ is the initial path then
8:              Continue
9:          end if
10:         if $t_i + GetSmoothedRTT(P_i) < t_{cur}$ then
11:             $P_s \leftarrow P_i$
12:             $t_{cur} \leftarrow t_i + GetSmoothedRTT(P_i)$
13:         end if
14:     end for
15:     $M_s \leftarrow M_s + W_s$    /* $W_s$ and $M_s$ are the CWND size of path $P_s$ and the amount of data should be assigned to path $P_s$ respectively */
16:     $S \leftarrow S - W_s$
17:     Path $s$ makes change to its CWND accordingly
18:     $t_s \leftarrow t_s + GetSmoothedRTT(P_s)$
19: end while
20: for $i = 0; i < n; i++$ do
21:     Restore congestion control parameters of each path
22: end for
23: Assign data to paths according to $M$
```

are two heterogeneous paths with smoothed RTT of $10ms$ and $35ms$ respectively. The triangle denotes ACK packets received on Path 1 and the square denotes ACK packets received on Path 2. The algorithm presumes data transmission starts on both paths from the beginning.

During the first $30ms$, because $t_2 + sRTT_2$ is always greater than $t_1 + sRTT_1$, the amount of data that should be assigned to Path 1 increases by the CWND of Path 1 at each time point. After that, $t_2 + sRTT_2 < t_1 + sRTT_1$ (i.e., $35ms$<$40ms$), so Path 2 is selected. The amount of data that should be assigned to Path 2 increases to the CWND of Path 2. The CWND of Path 2 is later changed according to the CCA. The calculation continues in the previously described manner until at $80ms$ when the basic scheduling unit is reached. Then, data can be assigned to paths according to this simulation process.

3.4. Discussions

3.4.1. How does ETC Handle Path Dynamics?

ETC uses the smoothed RTT of each path, which is maintained by the Path Manager of MPQUIC. Smoothed RTT is calculated with Exponentially Weighted Moving Average (EWMA). It therefore reflects an estimation of the path's end to end delay, which mitigates the impact of jitters.

Besides, the basic scheduling unit concept adopted in ETC also helps the scheduler adapt to path dynamics. After the

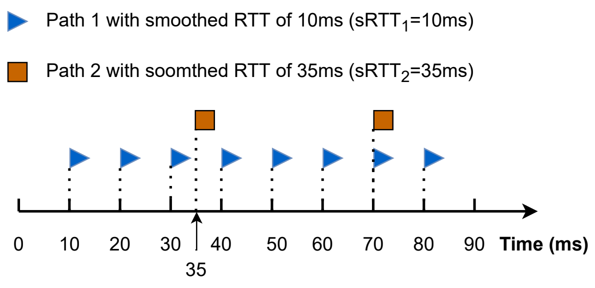

Path 1 with smoothed RTT of 10ms (sRTT1=10ms)
Path 2 with soomthed RTT of 35ms (sRTT2=35ms)
0
10
20
30
35
40
50
60
70
80
90
Time (ms)

Figure 4: An example of the simulation process

amount of data sent since the last computation reaches the size of a basic scheduling unit, ETC will calculate a new assignment using newly collected path information. That is to say, ETC updates the parameters used in transfer simulation for each basic scheduling unit. Hence, the size of a basic scheduling unit can be determined according to actual path dynamics. The rescheduling mechanism is another important way of dealing with path dynamics in ETC. The signal for the adopted CCA to detect congestion also works as the signal for rescheduling. It can be packet loss for loss-based CCA like RENO, CUBIC and OLIA.

3.4.2. Complexity Analysis for ETC Algorithm

The assignment of a basic scheduling unit is a sub-problem whose optimal solution is contained in the optimal solution to the entire problem, i.e., the problem exhibits optimal substructure. ETC takes congestion state change during the transferring of each basic scheduling unit into consideration, approximating the local optimal solution for each basic scheduling unit.

Besides, the Algorithm 1 contains a nested loop which accounts for its time complexity. The inner loop repeats n, the number of paths, times. The number of times the outer loop repeats is determined by the growth of paths' congestion windows. As congestion windows are always above zero, the algorithm is bound to converge. Assume the outer loop repeats K times, the time complexity of this algorithm is $O(Kn)$.

3.4.3. Analysis for Transfer Time With ETC

In the following, we will use a fluid model approximation to analyze the transfer time of ETC for sending a basic scheduling unit S. ETC involves simulating CWND changes on underlying paths, which is determined by CCA. Suppose the OLIA algorithm is used (ETC can also work with other CCAs in practice). For simplicity, we also assume that there is no congestion detected, thus the rescheduling mechanism will not be triggered.

Under OLIA, for each ACK received on the path P_i, its CWND W_i will be increased by

$$\frac{W_i/{rtt_i}^2}{(\sum_{P_j\in\mathbb{P}} W_j/rtt_j)^2} + \frac{\alpha_i}{W_i}, \tag{2}$$

where rtt_j is the round trip time of path P_j, $\mathbb{P}$ is the set of all available paths and α_i is a factor that guarantees responsiveness and nonflappiness.

Similar to [13], we have the differential equation

$$\Delta_i = \frac{dx_i}{dt} = \frac{x_i^2/{rtt_i}^2}{(\sum_{P_j\in\mathbb{P}} x_j)^2} + \frac{\alpha_i}{rtt_i^2}, \tag{3}$$

where x_i and x_j is the sending rate on path P_i and path P_j respectively. Equation(3) is derived with a fluid approximation of OLIA in which the stochastic variations of rates by their expectation are replaced. Thus, the amount of data sent during time t with OLIA is

$$\Phi(t,i) = \frac{1}{2}\Delta_i t^2. \tag{4}$$

When transfer time estimation begins, assume the CWND for path P_i obtained by ETC from OLIA sender is $W_{begin,i}$ and the slow start threshold is $SSTresh$. For simplicity, we only consider conventional slow start here. Path P_i will exit slow start at

$$t_{ess,i} = \log_2 \frac{SSTresh}{W_{begin,i}}. \tag{5}$$

Assuming the initial window is W_{init}, the amount of data transferred during slow start on path P_i can be estimated by

$$\varphi_i = \frac{2^{t_{ess,i}+1}}{t_{ess,i}+1} - \frac{2W_{begin,i}}{(\log_2 W_{begin,i} - \log_2 W_{init} + 1)W_{init}}. \tag{6}$$

The estimation stops when estimated amount of transferred data over all paths equals to basic scheduling unit S, i.e.,

$$S = \sum_{P_i\in\mathbb{P}} (\frac{1}{2}\Delta_i(t^2 - t_{ess,i}^2) + \varphi_i). \tag{7}$$

Hence, the transfer time t to send a basic scheduling unit S with ETC is

$$t = \sqrt{\frac{2S + \sum_{P_i\in\mathbb{P}}(\Delta_r t_{ess,i}^2 - 2\varphi_i)}{\sum_{P_j\in\mathbb{P}}\Delta_i}}. \tag{8}$$

4. Implementation

We have implemented the ETC scheduler for MPQUIC [19] based on *quic-go*. The calculation of CWND change is related to the CCA adopted by a specific MPQUIC implementation. And path information like RTT is maintained by MPQUIC's **Path Manager**, which can be easily obtained. Hence, it is easy to implement ETC in MPQUIC with minor modifications to CCA. We modified functions of *getCongestionWindow*() and *maybeInceraseCwnd*() in the implementation of *olia_sender.go* to obtain the CWND size and trigger changes to the CWND of each path. The smoothed RTTs of paths can be easily obtained through the *path.rttStats.SmoothedRTT*() function of *path.go*, which contains the definition of the data structure of paths. The scheduling algorithm is implemented with a data partition calculator, a weighted round-robin scheduler. After the data partition is calculated, the round-robin scheduler utilizes it as the weight for each path accordingly to assign data over paths.

5. Evaluation

This section provides the performance evaluation of the scheduler. The LRF scheduler is used as the baseline in the evaluation.

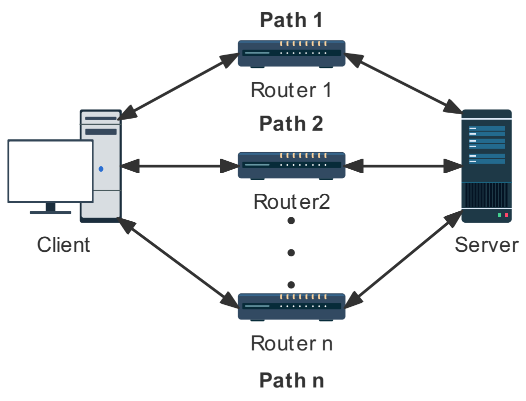


Figure 5: Experiment network topology

5.1. Experiment Setup

All experiments are performed in Mininet 2.2.2 on Ubuntu 14.04 LTS with Linux kernel 4.1.39. The host machine has Intel(R) Core(TM) i7-8700 3.20GHz CPU and 16GB RAM. Mininet runs on six cores with 4GB of RAM. Figure 5 illustrates the network topology adopted in the experiments.

The evaluation focuses on RTT and loss rate since they have profound impacts on network performance. To simulate heterogeneous paths, we categorize paths into paths with high/low latency, and paths with high/low loss rate. The thresholds of both latency and loss rate are defined according to Microsoft's documentation on media quality and network connectivity in its business communication platform [20] as MPQUIC has great potential for real time media. For any path whose RTT is greater than 100*ms*, it is deemed as a high latency path. To better observe the impact of path heterogeneity, the low latency paths' RTT is set to below 50*ms* in the evaluation. As of loss rate, any path whose loss rate is greater than 0.1% is considered to be a high loss rate path. According to Ookla's Speedtest Global Index [21], the global mean download speeds for mobile and fixed broadband are 73.50*Mbps* and 127.92 *Mbps* respectively (obtained in April 2022). Therefore, the bandwidth of paths is set to range from 90*Mbps* to 100*Mbps* to be in line with most of today's network connections. All network parameters (i.e., latency, loss rate and bandwidth) are all generated using the WSP algorithm [22] over ranges listed in Table 3. The experiment measures the time duration of retrieving a file from a Web server over multiple paths with randomly generated latency, loss rate and bandwidth. The http server and http client used in the experiment are from *quic-go* without modifications. The size of basic scheduling unit is set to 4*MB* in our evaluation.

Both short file download scenario and large file download scenario are evaluated over two heterogeneous paths, which is common such as mobile phone with both cellular and WiFi connectivity, PC with both fixed broadband and WiFi connectivity. As our evaluation mainly focuses on RTT and loss rate, these two attributes lead to four (2×2) path types. Considering these four path types, there are six (i.e., $C_4^2 = 6$) combinations of heterogeneous path groups (each path group consists of two heterogeneous paths with at least one different link characteristic) in total. Under each scenario, experiments are performed over all six possible types of heterogeneous path groups.

Table 3: Ranges of experimental parameters

	Low	**High**
RTT (ms)	(1, 50)	(100, 200)
Loss Rate (%)	(0, 0.1)	(0.1, 0.5)
Bandwidth (Mbps)	(90,100)	

For example, one of the path groups consists of a high latency & high loss path and a low latency & high loss path. This path group is denoted by a *hh_hl* label. The underline between each two characters separates different paths in a path group. The first character for each path describes its latency and the latter one stands for its loss rate. A *ll_lh* path group can represent multi-homed devices with both wired and wireless connections, since wireless connections are more likely to be disrupted.

The short file download scenario involves downloading 5MB of data over *h2quic* with TLS enabled. And for the large file download scenario, the amount of data is 100MB. The evaluation measures and analyzes the file transfer completion time under two scenarios using LRF, BLEST and ETC.

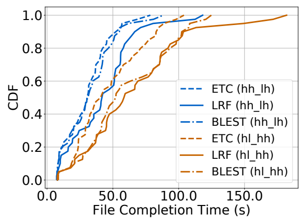
Besides, as the initial path for connection establishment may impact the performance of MPQUIC especially under the short file download scenario, we also compare performance when connection is established over each path in a path group respectively. The first and second path in each heterogeneous path group are referred to as Path #1 and Path #2 respectively. For example, for *hh_hl* path group, Path #1 (i.e., the path with high latency, high loss) is first used as the initial path to establish the connection. Then, Path #2 (i.e., the path with high latency and low loss) is used as the initial path for connection establishment.

Performance of three schedulers over more than 2 heterogeneous paths are also investigated under large file download scenario. The bandwidth of paths ranges from 90Mbps to 100Mbps and the loss rates of them are all set to low ($< 0.1\%$). The latency of paths is randomly and uniformly generated from high latency (100~200*ms*) or low latency (1~50*ms*). To ensure heterogeneity among paths, there are at least one low latency path and one high latency path in each path group.

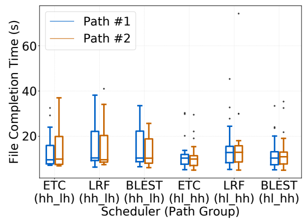
5.2. Results and Analysis

5.2.1. Large File Download

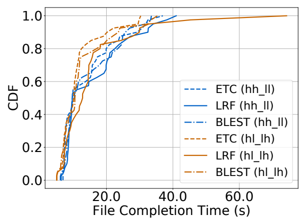
As shown in Figure 6, ETC stands out over all groups of heterogeneous paths. LRF often suffers from the long tail due to its transient metrics based decision making.



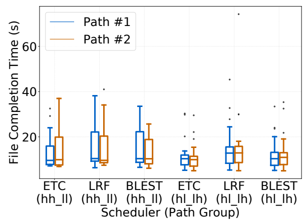
(a) Get 100MB over *hh_lh* and *hl_hh* path groups



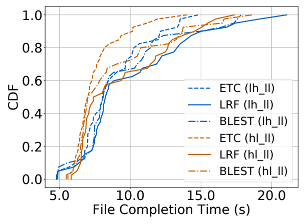
(b) Different initial path from *hh_lh* and *hl_hh* path groups



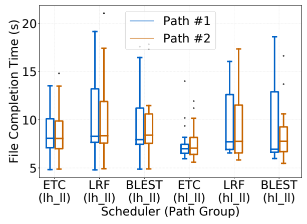
(c) Get 100MB over *hh_ll* and *hl_lh* path groups



(d) Different initial path from *hh_ll* and *hl_lh* path groups



(e) Get 100MB transfer over *lh_ll* and *hl_ll* path groups



(f) Different initial path from *lh_ll* and *hl_ll* path groups

Figure 6: Get 100MB using h2quic with TLS over six types of heterogeneous paths. ETC's performance is better than LRF and BLEST.

Figure 6a and Figure 6b demonstrate the performance of LRF, BLEST and ETC over the *hh_lh* and *hl_hh* path groups. In the *hh_lh* scenario, the high loss rate prevents the size of both paths' congestion windows from growing. When selecting a path, LRF tends to select the path with lower RTT first. But as the CWND size is limited, the total data to be sent in buffer cannot be assigned to the faster path at a time. At this time the faster path is occupied and deemed unavailable. As a result, LRF will select the path with higher RTT despite the fact that waiting for the faster path to be available again may lead to shorter transfer completion time. By avoiding blindly selecting paths with the LRF principle, BLEST shows much better performance. As for ETC, the data partition on each path is calculated in advance to make data transfer on both paths finish as simultaneously as possible. Consequently, the transfer time of all paths in this path group with LRF is about 28.7% longer than that with ETC.

As for performance over the *hl_hh* path group, ETC is approximately 29.6% faster than LRF. When the latency of the both paths are high, time taken by retransmission makes a difference in the overall file transfer completion time. Even though both paths are high in latency, LRF will still select a relatively low latency path as the prior path. If the prioritized path is high in loss rate, the retransmission constantly happens, resulting in performance degradation. BLEST does not have any mechanism to deal with packet loss, so its performance is close to LRF but without the long tail. ETC has higher responsiveness for packet loss as it works with OLIA in this simulation. The rescheduling mechanism of ETC ensures that the CWND change triggered by packet loss directly and timely impacts the data partition and the amount of data assigned to the path on which packet loss has happened is lowered. Hence, ETC shows the best performance.

Figure 6c and Figure 6d illustrate the performance over *hh_ll* and *hl_lh* path groups. Over the path group with a high latency, high loss path and a low latency, low loss path, the performance of all schedulers is close to each other. It works fine to prioritize a low latency path with fast growing CWND, so LRF shows satisfactory performance under this circumstance. Under the scenario with *hl_lh* path groups, the path with high latency is low in terms of loss rate while the loss rate of the path with low latency is high. In this case, LRF's strategy of blindly prioritizing the path with lower latency whose CWND keeps at a relatively small size incurs frequent retransmission. Also, the higher latency path is frequently selected due to the full window of faster path, which eventually leads to this severe long tail. On the other hand, BLEST can decide whether to wait for the faster path to be available again,so it successfully avoids the long tail. ETC schedules data on the high latency path and tries not to prolong the overall transfer time, i.e., the low latency path not waiting for the high latency path to finish transferring. Thus, ETC is about 26.9% faster than LRF in this experiment.

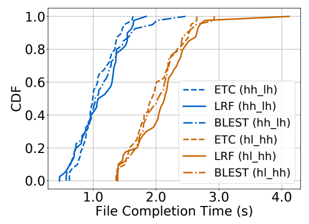
Figure 6e and Figure 6f depict two different path groups that both contains a low latency and low loss rate path. When both paths are low in latency. Their heterogeneity comes from their difference in loss rate. LRF still prioritizes a relatively low latency path and fails to deal with packet loss responsively. The same is also true for BLEST. As for ETC, even though the amount of data assigned to the path with relatively higher RTT and low loss rate may be quite small at first, it can be enlarged because of the path's growing CWND as the transmission goes by. Consequently, ETC achieves about 13.4% better performance than LRF. When both paths' loss rate is low (i.e., *hl_ll*), the overall transfer time of ETC is roughly 19.5% shorter than that of LRF. As LRF only takes available paths' latency into account to make scheduling decisions, it causes underutilization of underlying paths' capacity. Because BLEST also treats the slower path as a backup path, it underutilzes underlying paths' capacity as well. ETC, on the other hand, assigns data over all paths based on estimation before transmission to effectively shorten the overall transfer time.

5.2.2. *Short File Download*

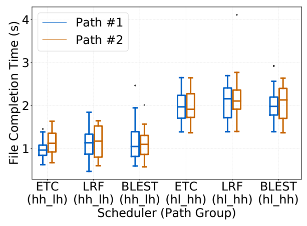
Figure 7 illustrates the short file download performance of the two schedulers over paths from the same path groups of the previous section. There is no significant difference between the performance of LRF, BLEST and ETC owing to the redundant sending strategy upon connection established. The initial path has great impact under all scenarios as the data to be transferred is only 5MB.

Under scenarios which involves a high latency path and a low latency path (*hh_lh*, *hh_ll*, *hl_lh*, *hl_ll*), a large amount of data has already been sent before the RTT of the high latency path is probed. The behavior of all schedulers doesn't play a significant part in influencing the overall file transfer completion time. Therefore, they basically show the same level of performance. However, the initial path shows great impact on the overall file transfer completion time.

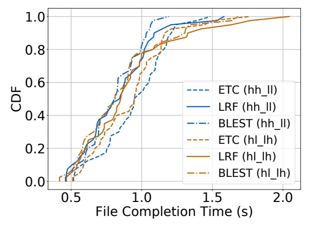
When it comes to path groups in which paths' latency is of the same level, i.e., both high latency or both low latency (*hl_hh*, *lh_ll*), there is no gap between the time of RTT probing over different paths. Consequently, the way of handling packet loss has the greatest impact on transfer completion time, which is elaborated in the previous section.



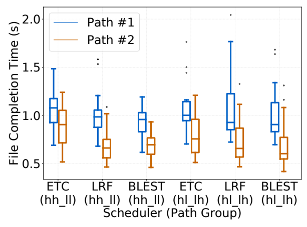
(a) Get 5MB over *hh_lh* and *hl_hh* path groups



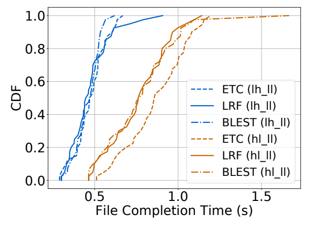
(b) Different initial path from *hh_lh* and *hl_hh* path groups



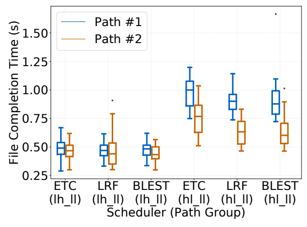
(c) Get 5MB over *hh_ll* and *hl_lh* path groups



(d) Different initial path from *hh_ll* and *hl_lh* path groups



(e) Get 5MB over *lh_ll* and *hl_ll* path groups



(f) Different initial path from *lh_ll* and *hl_ll* path groups

Figure 7: Get 5MB using h2quic with TLS over six types of heterogeneous paths.

5.2.3. *File Completion Time over Varied Number of Paths*

Figure 8 depicts the file completion time of 100MB over 2 to 5 heterogeneous paths. When the number of heterogeneous paths is two, it falls into the aforementioned *hl_ll* path group. As the number of heterogeneous paths increases from 2 to 3 (*hl_ll_ll*), the file transfer completion time decreases by about 34.4%, 38.8% and 39.9% respectively with ETC, BlEST and LRF. When the number of heterogeneous path comes to 4 (*hl_ll_hl_ll*), the median file transfer completion time barely shortens with LRF and slightly shortens by about 9.7% and 9.9% with ETC and BLEST respectively. When there are four low latency paths and one high latency path (*hl_ll_ll_ll_ll*), three schedulers' performance is close to each other. In a nutshell, ETC shows favorable performance and still outperforms LRF as the number of heterogeneous paths increases.

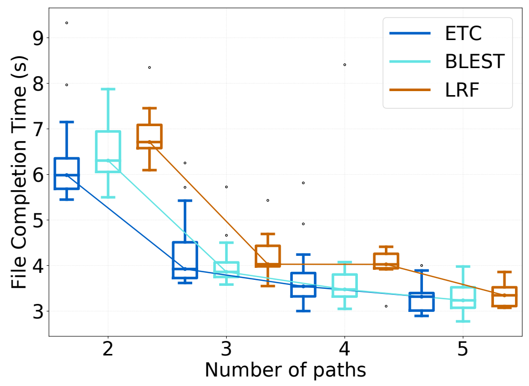


Figure 8: Performance on varied number of heterogeneous paths

6. Related Works

6.1. *Multipath Transport Protocols*

The idea of multipath transport protocol dates back to Stream Control Transmission Protocol (SCTP) [23] which is only deployed in certain fields like signaling traffic within telephone networks as a failover measurement. Not until the prevalence of multihomed mobile devices, does MPTCP [12] emerge as a multipath transport protocol that is possible to be widely deployed. After several years of evolving, schedulers of different kinds have proposed for it [24], making it more and more mature. However, due to ossification [25], MPTCP is still not prevailing.

With the emergence of QUIC, inspired by MPTCP, De Coninck et al. presented the design and evaluation of Multipath QUIC [3], which shows that MPQUIC has better multipath performance than MPTCP. The flexibility of MPQUIC also brings various shceduler for different purpose [26, 27, 28, 29] and makes it easy to transplant schedulers from MPTCP. As MPQUIC is UDP-based,its deployment does not involve any changes to operating systems and middleboxes. As a result, MPQUIC is a more promising multipath transport protocol.

6.2. *Multipath Over Heterogeneous Paths*

To deal with performance degradation and achieve low latency scheduling over heterogeneous paths, STTF [9] is proposed. The design of STTF is similar to Out-of-Order Transmission for In-Order Arrival Scheduler (OTIAS) [15] and Earliest Completion First (ECF) [16]. They all use the path information to estimate transfer completion time and selects the fastest path regardless of whether the CWND is full. One significant difference between STTF and other schedulers (e.g., OTIAS and ECF) is that it takes congestion state into consideration instead of assuming that the congestion state is congestion avoidance, which results in more accurate and practical transfer time estimation. Moreover, STTF adopts rescheduling mechanism. Whenever transmission is interrupted by external events, STTF will remove previous scheduling assignments and reschedule all unsent data. Therefore, STTF is capable of promptly reacting to changes of underlying network.

To achieve low latency scheduling, STTF will estimate the transfer time for every segment over each path first. Then the path with the lowest estimated transfer time is selected to transfer the segment. However, STTF's calculation and scheduling is on a per-segment basis, which is not suitable for MPQUIC. In current MPQUIC implementation, data is read in a way of streaming and the size of the data read at a time is quite small. Consequently, it cannot trigger the congestion window change. As a result, STTF falls back into LRF, failing to fulfill its purpose. In contrast with STTF, the goal of the ETC scheduler proposed in this paper is to achieve low overall file transfer completion time. What it does is to assign data to paths according to the assignment calculated in transfer simulation. Though ETC also involves transfer time estimation, the estimation is conducted with the input to be data blocks.

7. Conclusions

This paper investigates the performance degradation issue of MPQUIC over heterogeneous paths and presents a novel scheduler, called ETC, to address this issue. Taking the transfer completion time and congestion states of each path into consideration, ETC shortens the overall transfer completion time by up to approximately 29.6% compared with LRF in the evaluation. Moreover, the basic scheduling unit concept makes ETC more flexible and cost less computation resources. However, it may suffer from the side effect of a large basic scheduling unit for short file transfer, which needs to be solved in future work.

References

[1] D. Madariaga, L. Torrealba, J. Madariaga, J. Bermúdez, J. Bustos-Jiménez, Analyzing the adoption of QUIC from a mobile development perspective, in: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, 2020, pp. 35–41.

[2] V. S. Kubde, S. D. Sawarkar, Exploring multipath tcp schedulers in heterogeneous networks, International Journal of Information Communication Technologies and Human Development 14 (1) (2022) 1–11.

[3] Q. De Coninck, O. Bonaventure, Multipath QUIC: Design and evaluation, in: Proceedings of the 13th international conference on emerging networking experiments and technologies, 2017, pp. 160–166.

[4] 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 the 2021 ACM SIGCOMM 2021 Conference, 2021, pp. 418–432.

[5] M. Bishop, HTTP/3, RFC 9114 (Jun. 2022). doi:10.17487/RFC9114.
URL https://www.rfc-editor.org/info/rfc9114

[6] Z. Tan, J. Zhao, Y. Li, Y. Xu, S. Lu, Device-Based LTE latency reduction at the application layer, in: 18th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2021, pp. 471–486.
URL https://www.usenix.org/conference/nsdi21/presentation/tan

[7] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, T. Moscibroda, Characterizing and improving WiFi latency in large-scale operational networks, in: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 2016, p. 347–360. doi:10.1145/2906388.2906393.
URL https://doi.org/10.1145/2906388.2906393

[8] GSM Association, The Mobile Economy 2021 (2021).
URL https://www.gsma.com/mobileeconomy/wp-content/uploads/2021/07/GSMA_MobileEconomy2021_3.pdf

[9] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, N. Kuhn, Low-latency scheduling in MPTCP, IEEE/ACM Transactions on Networking 27 (1) (2018) 302–315.

[10] J. Iyengar, M. Thomson, QUIC: A UDP-Based Multiplexed and Secure Transport, RFC 9000 (2021). doi:10.17487/RFC9000.
URL https://www.rfc-editor.org/info/rfc9000

[11] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, M. Kühlewind, Multipath Extension for QUIC, Internet-Draft draft-ietf-quic-multipath-01, Internet Engineering Task Force, work in Progress (2022).
URL https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-01

[12] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, C. Paasch, TCP Extensions for Multipath Operation with Multiple Addresses, RFC 8684 (2020). doi:10.17487/RFC8684.
URL https://www.rfc-editor.org/info/rfc8684

[13] R. Khalili, N. Gast, M. Popovic, J.-Y. Le Boudec, MPTCP is not Pareto-optimal: Performance issues and a possible solution, IEEE/ACM Transactions on Networking 21 (5) (2013) 1651–1665.

[14] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, G. Smith, Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer, in: 27th International Conference on Advanced Information Networking and Applications Workshops, IEEE, 2013, pp. 1119–1124.

[15] F. Yang, Q. Wang, P. D. Amer, Out-of-order transmission for in-order arrival scheduling for multipath TCP, in: 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE, 2014, pp. 749–752.

[16] Y.-s. Lim, E. M. Nahum, D. Towsley, R. J. Gibbens, ECF: An MPTCP path scheduler to manage heterogeneous paths, in: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, 2017, pp. 147–159.

[17] S. Ferlin, Ö. Alay, O. Mehani, R. Boreli, BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks, in: IFIP Networking Conference and Workshops, IEEE, 2016, pp. 431–439.

[18] M. Drozdowski, Scheduling for parallel processing, Vol. 18, Springer, 2009.

[19] Q. D. Coninck, qdeconinck/mp-quic, original-date: 2017-09-15T11:19:20Z (2022).
URL https://github.com/qdeconinck/mp-quic

[20] Media Quality and Network Connectivity Performance - Skype for Business Online | Microsoft Docs (2021).
URL https://docs.microsoft.com/en-us/skypeforbusiness/optimizing-your-network/media-quality-and-network-connectivity-performance

[21] Speedtest Global Index – Internet Speed around the world – Speedtest Global Index (Apr. 2022).
URL https://web.archive.org/web/20220419065239/https://www.speedtest.net/global-index

[22] J. Santiago, M. Claeys-Bruno, M. Sergent, Construction of space-filling designs using WSP algorithm for high dimensional spaces, Chemometrics and Intelligent Laboratory Systems 113 (2012) 26–31.

[23] R. R. Stewart, M. Tüxen, M. Proshin, Stream Control Transmission Protocol: Errata and Issues in RFC 4960, RFC 8540 (2019). doi:10.17487/RFC8540.
URL https://www.rfc-editor.org/info/rfc8540

[24] B. Y. Kimura, D. C. Lima, A. A. Loureiro, Packet scheduling in multipath TCP: Fundamentals, lessons, and opportunities, IEEE Systems Journal (2020).

[25] K. Edeline, Characterizing and modeling of transport-based middleboxes,

Ph.D. thesis, Université de Liège, Liège, Belgique (2019).

[26] M. M. Roselló, Multi-path scheduling with deep reinforcement learning, in: European Conference on Networks and Communications, IEEE, 2019, pp. 400–405.

[27] R. S. Mogensen, C. Markmoller, T. K. Madsen, T. Kolding, G. Pocovi, M. Lauridsen, Selective redundant MP-QUIC for 5G mission critical wireless applications, in: IEEE 89th Vehicular Technology Conference, IEEE, 2019, pp. 1–5.

[28] A. Rabitsch, P. Hurtig, A. Brunstrom, A stream-aware multipath QUIC scheduler for heterogeneous paths, in: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, 2018, pp. 29–35.

[29] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, G. Caso, Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments, IEEE Journal on Selected Areas in Communications 38 (10) (2020) 2295–2310.