PStream: Priority-Based Stream Scheduling for Heterogeneous Paths in Multipath-QUIC

Xiang Shi*, Lin Wang†‡, Fa Zhang*, Biyu Zhou§ and Zhiyong Liu*
*Institute of Computing Technology, Chinese Academy of Sciences, China
†Vrije Universiteit Amsterdam, The Netherlands
‡Technische Universität Darmstadt, Germany
§Institute of Information Engineering, Chinese Academy of Sciences, China
Email: shixiang@ict.ac.cn, lin.wang@vu.nl, zhangfa@ict.ac.cn, zhoubiyu@iie.ac.cn, zyliu@ict.ac.cn

***Abstract*—Web latency remains the main obstacle to improving user experience with the continuous development of the web. A lot of works have been made in this course. Quick UDP Internet Connection (QUIC) embeds stream multiplexing to solve the head-of-line blocking caused by the in-order requirement of TCP. Multipath-QUIC (MPQUIC) brings further improvements by utilizing multiple paths, as is done in MultiPath TCP (MPTCP). Different from MPTCP schedulers, MPQUIC schedulers are stream-aware and thus can provide finer granularity of multipath scheduling. As streams are with different features based on their contents, the resource preferences of a stream are highly related to its feature. We find that scheduling without the recognition of the stream features can aggravate inter-stream blocking when sharing paths. We fill this gap and propose PStream – a priority-based online stream scheduling mechanism for MPQUIC, which performs path scheduling based on the stream features. We examine the effectiveness of PStream under different path heterogeneity comparing to the original and the latest scheduler of MPQUIC. Our evaluation shows that our scheduler can reduce up to 25.4% of page load time in high path heterogeneity.**

***Index Terms*—QUIC; MPQUIC; stream scheduling; prioritization; web latency**

I. Introduction

Over the decades, web pages have undergone tremendous changes. The richness and complexity of web contents and the number of domains used to retrieve these contents continue to grow rapidly. In this situation, reducing web latency has become a crucial issue to improve user-perceived performance [1].

Protocols have been improving to reduce web delay. One of the major issues is to solve the head-of-line (HOL) blocking problem due to restrictions on the transmission order – a single slow transmission can block all the transmissions behind it [2]. To solve the HOL blocking due to the in-order transmission requirement of TCP, Quick UDP Internet Connection (QUIC) was proposed to provide stream multiplexing [3] in the transport layer. In a QUIC connection, an application-oriented stream is provided for each HTTP request/response, guaranteeing that a late packet for each individual stream will not block other concurrent streams.

Another issue is the prioritization of web content. A request for time-critical resources can be blocked by non-critical ones when sharing a path with limited bandwidth. In the application layer, HTTP/2 [4] recognizes this problem and performs prioritization on HTTP requests. Each request is assigned a priority that represents the proportion of resources allocated to it. A recent study [5] investigates the impact of HTTP/2 prioritization and confirms that prioritization improves web performance after careful alignment with other components of the webpage load process.

In the transport layer, prioritization of web content can be achieved with finer granularity in QUIC and its variant as stream-multiplexed protocols. Multipath-QUIC (MPQUIC) [6] [7], one variant of QUIC, was proposed aiming at enabling handover and aggregating the bandwidth of multiple paths. The performance of a multipath protocol can be highly affected by the mechanism of scheduling packets over multiple paths. As a result, works have been done to investigate into the packet scheduling of MPQUIC. Different from MultiPath-TCP (MPTCP) schedulers [8], MPQUIC schedulers are stream-aware and thus can support different applicational needs.

After investigating into the existing MPQUIC schedulers, we observe two aspects to be improved. The first is about the burst transmission pattern [8]. Packets of a stream first compete for the fast path and then use the slow path after the fast path is temporarily unavailable. The burst sending of packets increases out-of-order arrival and buffer requirements of both host and in-network [8]. As a result, packets that exceed the buffer capacity have to be dropped, resulting in the throughput degradation on the fast path. Moreover, the late packets on the slow path can delay the overall completion time of the stream. The key way to alleviate this condition is to reduce the out-of-order arrival at the receiver.

The second is about the recognition of stream feature in path scheduling. One feature is the priority. Since the priority represents the time-criticality of the streams, not incorporating HTTP/2 prioritization into the schedulers can result in suboptimal performance [9], especially for time-critical

X. Shi is also with University of Chinese Academy of Sciences, Beijing, China. The Corresponding Author is Zhiyong Liu (zyliu@ict.ac.cn). This work is partially supported by the National Natural Science Foundation of China (grant numbers 61520106005, 61761136014) and the National Key Research and Development Program of China (grant number 2017YFB1010001). L. Wang was funded partially by the German Research Foundation (DFG) under grant 392046569 and by the DFG Collaborative Research Center (CRC) 1053 – MAKI.

streams. In addition to the priority, the size feature should also be carefully considered by the schedulers. We find that the resource preference of a stream is highly related to its size feature. For example, big streams prefer paths with higher bandwidth, while small streams favor paths with lower RTT [5]. If streams with different size features share the same path, small streams can be influenced by long-lived and bandwidth-consuming big streams.

In this paper, we propose an online packet scheduler – PStream for MPQUIC. Instead of making all streams competing for the fast path in a greedy fashion, we allocate paths for each stream by considering the match of stream and path features in the scheduling process. The streams can utilize the allocated paths concurrently from the beginning, thus reducing the burst transmission on the fast path.

Concretely, we make the following contributions: 1) We summarize the basic principles in priority-based stream scheduling and devise an optimization model for the scheduling problem. 2) We propose a priority-based stream scheduling mechanism, namely PStream for MPQUIC. 3) We implement PStream based on the MPQUIC open-source implementation [6] and carry out experiments to validate the basic properties of PStream. We evaluate its performance comparing to the original (LRF) and the latest (SA-ECF) scheduler of MPQUIC and analyze the results.

The rest of the paper is organized as follows: Section II introduces the background and Section III presents the related work. Section IV describes the design rationale of PStream and discusses its implementation. Section V analyzes the experimental results. Section VI concludes the paper and discusses future work.

II. BACKGROUND

A. HTTP/2

In HTTP/2, a stream represents an independent, bidirectional sequence of frames exchanged between the client and server within an HTTP/2 connection [4]. A client (e.g. a browser) can assign priority preferences for a new stream to express how it would prefer its peer to allocate resources for the stream when managing concurrent streams. An HTTP/2 prioritization strategy is the strategy of priority preference assignment for all the streams. Different browsers adopt widely different HTTP/2 prioritization strategies [5]. The prioritization strategy includes two aspects. The first is called stream dependency – streams can be prioritized by marking them as dependent on the completion of other streams. The other is a relative weight (used as the priority value in this paper) given to each stream that indicates the proportion of resources for the stream. The prioritization strategy can be presented by a dependency tree. We use the dependency tree in Fig. 1 as an illustrating example. Streams A, B and C are initiated concurrently and share resources proportionally based on the assigned priority values. Stream D and E cannot be initiated until A is completed. When A completes transmission before B and C, the newly initiated stream D and E continue to share resources proportionally with B and C based on their priorities.

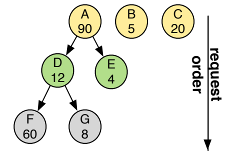


Fig. 1. An example of the HTTP/2 dependency tree. The streams are initiated in request order and share resources based on the priorities.

Each dependent stream is allocated a priority value between 1 and 255 (inclusive) and a higher value means a higher priority.

B. QUIC and MPQUIC

QUIC is an encrypted, multiplexed, and low-latency transport protocol which is designed from ground up to replace most of the traditional HTTPS stack: HTTP/2, TLS, and TCP. QUIC is designed to reduce web latency, with the following main features. First, QUIC is developed as a user-space transport that enables fast iterative changes. QUIC uses UDP as a substrate to allow QUIC packets to traverse middleboxes. Consequently, the ossification caused by middleboxes [10] [11] when deploying different QUIC versions is avoided. Second, QUIC minimizes handshake delay by combining cryptographic and transport handshakes. Third, to reduce HOL blocking delay brought by the sequential delivery of TCP, QUIC multiplexes multiple requests/responses over a single connection by providing each with a stream. In QUIC, a single packet can contain frames of one or more streams. Streams encapsulate their data into frames, and then these frames are bundled into QUIC packets. Therefore, lost of a QUIC packet only blocks streams with data contained in this packet.

Experience from Multipath TCP (MPTCP) [12] reveals that exploiting different paths between a client and a server utilizes different paths and enables faster handovers especially in mobile devices, which improves user experience. QUIC cannot support this for now because it uses a single UDP flow between the client and the server. Therefore, Multipath QUIC (MPQUIC) [6] filled this gap by enabling QUIC to use several paths simultaneously. MPQUIC uses a path manager to control the creation and deletion of paths. The information of the paths is shared between two end hosts and is periodically updated. Due to the increased complexity of computation and maintenance, it has been shown that increasing the number of paths provides diminishing returns [13]–[16], and the bulk of benefits of multipathing can be reaped with only two paths [14].

III. RELATED WORK

Works have been done to investigate into the packet scheduling of MPQUIC. The original mechanism of MPQUIC [6] disregards stream priority and packs a packet using a round-robin mechanism that cycles through the streams. Then for each packed packet, MPQUIC adopts the Lowest-RTT-First (LRF) scheduler, which uses round-trip time (RTT) as the only criterion for path scheduling. However, HTTP/2 prioritization is not incorporated into the LRF scheduler and failing to account for priorities of streams can result in suboptimal performance [9], especially for time-critical streams. In addition, other path characteristics like bandwidth should also be included when considering path heterogeneity [8].

SA-ECF [17] introduced HTTP/2 prioritization into the scheduler by applying the weighted Round Robin (WRR) algorithm for the streams. To schedule a path for a packet, SA-ECF first finds two paths with the lowest RTT and then makes decisions between the two paths. The decision is based on the completion time estimation of the stream that the packet belongs to. However, the estimation uses the remaining bytes of the stream instead of the stream size. As a result, packets of different streams compete for the fast path in a greedy fashion without the full picture of the stream size, causing burst transmission [8] on the fast path.

[18] considers both stream size and priority features in scheduling. To further optimize the completion time of the time-critical stream, it schedules a stream to a single path. This sacrifices the bandwidth aggregation benefits of multipath transmission. [19] assumes the dependency tree of a web page is known in advance at the server-side and then uses this information to optimize the transmission order of all the streams. However, the assumption is not practical for online cases. Morgensen et al. [20] consider reliability as the major design goal and present a scheduler for MPQUIC where data are scheduled redundantly over multiple paths for real-time vehicular communication in LTE networks, while our major goal is to reduce web latency by prioritized scheduling.

IV. PStream Design and Implementation

PStream is an online solution for stream scheduling in MPQUIC to reduce web latency. Particularly, it performs path scheduling based on the stream features and is incorporated with HTTP/2 to provide prioritization down to the transport layer.

A. An Example

An example of PStream is depicted in Fig. 2. In the application layer, a client sends two requests to the web server with each assigned a priority value, representing the time-criticality of the request. Higher priority of a request means the client expects it to complete earlier. Upon receiving the requests, the server initiates two streams for each request. For the concurrently initiated two streams, PStream first schedules stream A with the higher priority to path #1. Then it schedules stream B to both paths for bandwidth aggregation. The scheduling decisions are based on the completion time estimation of the stream. Resources are shared proportionally to priorities among the streams on a single path (i.e. path #1).

B. Overview

Fig. 3 illustrates the main structure of a PStream connection. A connection is established after the cryptographic handshake of MPQUIC and is identified by a Connection ID (CID). There can be multiple available paths within a connection. Each path is scheduled from none to multiple streams and

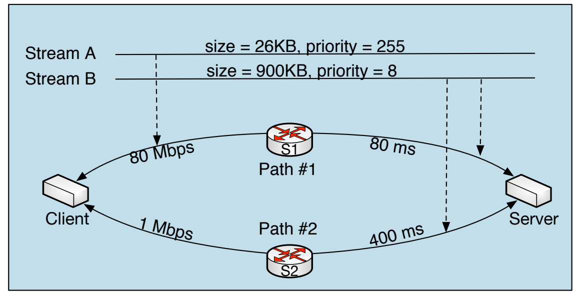


Fig. 2. An example of PStream scheduling two concurrently initiated streams: the streams are scheduled following the descending priority order.

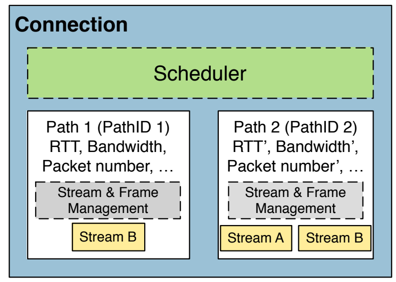


Fig. 3. An overview of a PStream connection. The blocks with dash-line borders show our modifications on MPQUIC [6].

has a stream manager that manages the transmission of the streams. A global scheduler is responsible for the stream-to-path scheduling based on our proposed scheduling policy.

C. Stream and Path

A stream is a lightweight bi-directional byte stream abstraction for the transmission of an HTTP request/response. Each stream has a unique stream ID and an HTTP/2 priority value that indicates the time-criticality of the stream. Streams can be opened freely after the QUIC connection establishment, and can be canceled or closed without tearing down the entire connection. Each path has a unique path ID to differentiate from each other. The RTT and bandwidth of a network path can be measured and periodically updated [21], [22]. We maintain smoothed estimations of RTT and bandwidth for each path during the transmission.

D. Scheduler

The scheduler is designed to schedule streams to paths. Note that as the stream dependencies [4] restrict the initiation order of streams, we consider the path scheduling of concurrently initiated streams each time. Upon the initiation of a stream, the scheduler makes the scheduling decision for the stream and schedules streams on the selected paths. To come up with an effective scheduler, we observe the following principles to be followed by the scheduling policy.

- **PRCP-1** Concurrent streams should be scheduled in descending order of priority.
- **PRCP-2** The completion time of a stream on each path should be balanced to minimize the overall stream completion time.

TABLE I
SCHEDULING PARAMETERS

Symbol	Meaning
D	total data volume of stream
pr	the priority of the stream
T	overall completion time of stream
i	path i
n	number of paths
o_i	estimated one-way delay of the path i
b_i	estimated bandwidth of the path i
d_i	fraction of data assigned to path i
bs_i	bandwidth share of stream on path i
sum_i	priority sum of the streams on path i
t_i	completion time of path i

- **PRCP-3** Resource should be shared based on priorities for streams on the same path.

To satisfy PRCP-1, the scheduler schedules the streams in descending order of priority for concurrent streams. To satisfy PRCP-2, we derive an optimization model for the path scheduling of one stream. Since the overall completion time of a stream can be prolonged by the slowest path, packets of the stream should be scheduled properly to balance its completion time on heterogeneous paths. For the stream s, we calculate the distribution of its data on each path that satisfies the shortest estimated completion time. We first define T for the overall completion time of the stream and t_i for the estimated completion time on each path. n is the number of paths. Then we introduce D for the data volume of stream s and define d_i for the data volume scheduled on path i. Therefore, t_i can be calculated by

$$t_i = \begin{cases} 0, & d_i = 0 \\ \dfrac{d_i}{bs_i} + o_i, & d_i > 0 \end{cases} \tag{1}$$

where o_i is the estimated one-way delay of the path i and bs_i is the bandwidth share assigned to stream s if stream s is scheduled to path i. To satisfy PRCP-3, we design a bandwidth sharing mechanism for streams on the same path. We suppose the sum of priority of all the streams that are already scheduled to path i is sum_i, and the priority value of stream s is pr, and then bs_i can be calculated by:

$$bs_i = \frac{pr}{sum_i + pr} \cdot b_i \tag{2}$$

where b_i is the smoothed bandwidth estimation of the path i. Then the scheduling problem can be summarized and formulated as follows:

$$\begin{aligned} &(P_1) \qquad \min T \\ &\text{subject to} \\ &\qquad t_i = \begin{cases} 0, & d_i = 0 \\ \dfrac{d_i}{bs_i} + o_i, & d_i > 0 \end{cases} \\ &\qquad \textstyle\sum_{i=1}^{n} d_i = D \\ &\qquad T = \max\{t_i \mid i \in [1, 2, ..., n]\} \end{aligned} \tag{3}$$

To solve the model, we propose an online algorithm and the pseudo-code is illustrated in Algo.1, which is composed of a procedure of three steps.

Algorithm 1 PStream algorithm

procedure SCHEDULESTREAM
  sortedList $\leftarrow$ sort paths by ascending order of o_i
  initialize $d_i = 0$, $i \in [1, 2, ..., n]$
  $D' \leftarrow D$
  for path i in $sortedList[1, ..., n-1]$ **do** ▷ Fill the gap
    $o_{gap} \leftarrow o_{i+1} - o_i$
    if $o_{gap} \neq 0$ and $D' > 0$ **then**
      if $D' >= o_{gap} \times \sum_{k=1}^{i} bs_k$ **then**
        for $j \leftarrow 1$ to i **do**
          $inc \leftarrow o_{gap} \times bs_j$
          $d_j \leftarrow d_j + inc$
          $D' \leftarrow D' - inc$
        end for
      else
        for $j \leftarrow 1$ to i **do**
          $inc \leftarrow D' \times (bs_j / \sum_{k=1}^{i} bs_k)$
          $d_j \leftarrow d_j + inc$
        end for
        $D' \leftarrow 0$
      end if
    end if
  end for
  if $D' > 0$ **then** ▷ Share the rest volume proportionally
    for path i in *sortedList* **do**
      $d_i \leftarrow d_i + D' \times (bs_i / \sum_{k=1}^{n} bs_k)$
    end for
  end if
end procedure

Step 1: Since the minimum of completion time on each path is determined by the one-way delay o_i, we sort paths by the ascending order of o_i.

Step 2: The *fillgap* process schedules data to balance the completion time gap of multiple paths. We start the process from $path_1$ with the smallest one-way delay o_1, and compare o_1 with the second smallest one-way delay o_2. We schedule data of $(o_2 - o_1) \times bs_1$ on $path_1$ to make the completion time on $path_1$ to equal o_2. Then we compare o_2 and o_3. We fill the gap through scheduling data of $(o_3 - o_2) \times bs_1$ on $path_1$ and data of $(o_3 - o_2) \times bs_2$ on $path_2$. The process is repeated until all the data is allocated or the completion time of all the paths becomes equivalent.

Step 3: Third, the remaining data are allocated in proportion to the path bandwidth to make the completion time balanced on multiple paths.

The time complexity of this algorithm is $O(n^2)$. In real networks, due to the increased complexity of computation and maintenance, the number of paths is limited. Meanwhile, the algorithm is executed only once for each stream. In our simulations, we did not observe any noticeable time overhead brought, and this also applies to real networks.

E. Implementation

To validate the effectiveness of our proposal, we complete a preliminary implementation based on existing MPQUIC open-source implementation written in Go [6]. The code is open-sourced at [23]. The details of PStream implementation are elaborated in the following parts.

Overview. The scheduler is triggered every time a new stream arrives. Based on the proposed algorithm, the scheduler allocates the new stream to each path with a calculated amount of data. After the scheduling process, the stream is added to the corresponding paths for sending.

Sending loop. The transmission is performed after the arrived streams have been scheduled to paths. The whole transmission process is composed of repeated transmission rounds. In each transmission round, paths get sending opportunities equal to the number of streams on the path. This number is updated when a stream is newly added to the path or closed after its transmission completed. If the congestion window of a path is full or no streams are available on this path, we skip it and go to the next path. The transmission round is repeated until all streams finish sending.

Path scheduling. The estimated path parameters are needed for scheduling a stream to paths. As a starting point, we set up initial values of RTT and bandwidth based on prior knowledge on creating a path; other techniques can also be used [21], [22]. During the transmission, we maintain smoothed values of RTT estimation of each path. The RTT estimation is based on the exponential weighted moving average mechanism of TCP [21]. And the bandwidth estimation is based on [24].

Stream scheduling. Stream scheduling refers to the mechanism to transmit streams on the same path. The implemented mechanism is divided into two aspects. First, to achieve the proportional bandwidth sharing mechanism of streams, we let the path select only one stream at a time for packet packing. The probability of a stream being selected is calculated by dividing its priority by the priority sum of all the scheduled streams on this path. Second, since the amount of data a stream is assigned to a path is limited and decided by the scheduler, it is not ideal that this number can align with the packet boundaries. If we bundle the leftover bytes of different streams together into one packet, then the loss of this packet can affect both streams. Therefore, we pack data of a single stream into one packet, and the last packet of a stream may contain a relatively small amount of data.

V. Experiments

We evaluate our mechanism PStream with comparison to the LRF scheduler [6] and the SA-ECF scheduler [17] on the Mininet emulation platform [25]. To provide a fair assessment of the compared implementations, we adopt an experimental design similar to the one used for MPQUIC [6]. We evaluate the stream completion time and page load time under different heterogeneity scenarios for available paths.

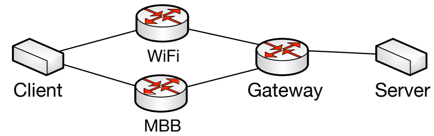


Fig. 4. A typical multipath scenario over two network paths, which we used for our evaluation.

A. Experimental Setup

Network. We consider the typical network topology with two multi-homed hosts over disjoint paths used by [6] [17], as depicted in Fig. 4. This topology allows us to evaluate multipath-scenarios like a mobile phone connecting to both Mobile BroadBand (MBB) and WiFi at the same time [17]. To explore the performance impact of various path heterogeneity, we set bandwidth and RTT heterogeneity for the two paths. The maximal receive window is set to 16 MB. Packet losses caused by router buffer overflow can occur in these environments. We perform our evaluation on a laptop with an Intel Core i5 Dual-Core CPU i5-4278U@2.60GHz with 8.0GB RAM.

B. Baseline Schedulers

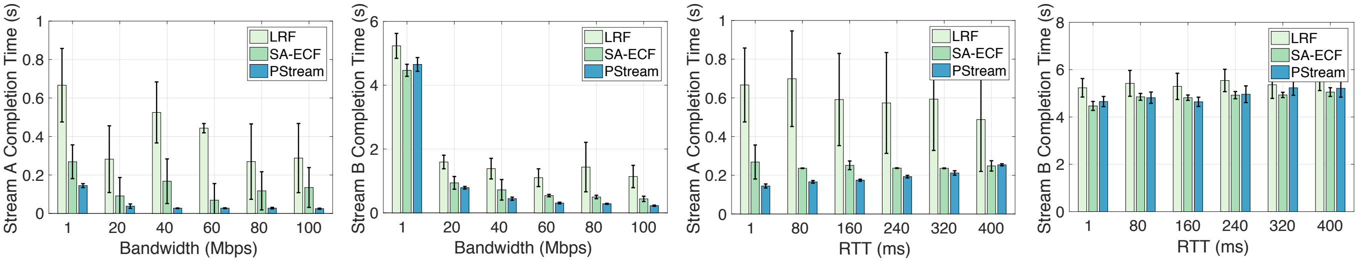
Lowest-RTT-First (LRF). LRF [6] round-robinly packs data of all the streams for a single packet. Upon sending the packet, an LRF sender needs to select a path for it. It maintains a smoothed round-trip time (RTT) estimation of each path. Each time sending a packet, it prefers the lowest RTT path with an available congestion window.

SA-ECF. SA-ECF [17] provides a packet scheduler to avoid transmission delayed by slower paths. For each packet, SA-ECF first chooses two different paths: One path is with the lowest RTT and not necessarily to be an available path, while the other is an available path with the lowest RTT. Then SA-ECF estimates the completion time of the stream to which the packet belongs on two paths, and schedules the packet to the path with the lowest estimated time. The number of opportunities for each stream to send is proportional to its weight.

C. Stream-level Evaluation

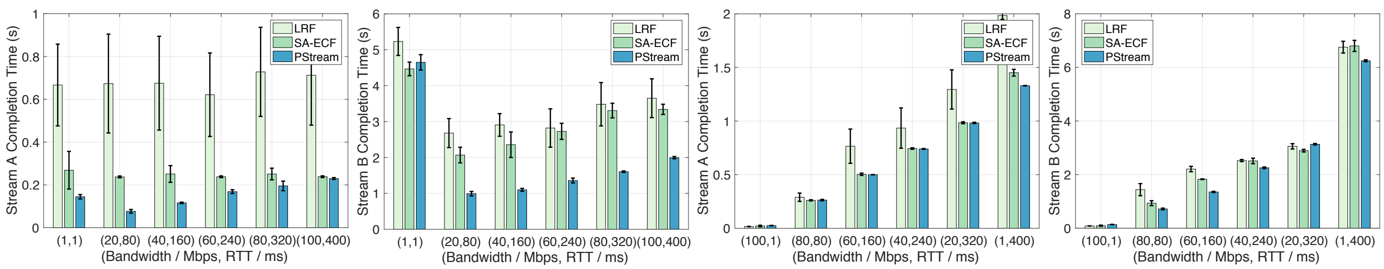
In this experiment, we cover a full range of path heterogeneity to explore the impact of different schedulers on the completion time of the two streams with diverse resource preferences. Note that the completion time of a stream represents the start from the request to the completion of the stream. Reducing the stream completion time can speed up the page loading process in the browser. For example, earlier completed streams such as some key requests (HTML, CSS) can allow browsers to parse earlier. Therefore, reducing the stream completion time can speed up the loading speed of the whole web page and benefit the user's experience.

Traffic. We evaluate the completion time of two concurrently initiated streams corresponding to two types of web objects inside a web page. This prioritization strategy (i.e. priority assignment and the order of initiating streams) is based on the weighted round-robin mechanism of Safari 11 [5]. The



(a) (b) (c) (d)

Fig. 5. The impact of path bandwidth and RTT heterogeneity on the completion time: (a) and (b) are results of stream A and B respectively under bandwidth heterogeneity, (c) and (d) are results of stream A and B respectively under RTT heterogeneity. Each repeated simulation carries a standard deviation value.



(a) (b) (c) (d)

Fig. 6. The impact of changing both bandwidth and RTT of path #2 at the same time: (a) and (b) are results of stream A and B respectively when each path has its advantage, (c) and (d) are results of stream A and B respectively when one path wins over the other. Each repeated simulation carries a standard deviation value.

first is a time-critical small stream A corresponding to an HTML document, with 26 KB size and priority value to be 255. The second is a non-critical big stream B corresponding to an image, with 900KB size and priority value to be 8. The client measures the completion time of a stream by recording the time elapsed between the initiation of the request and the reception of the last byte of the corresponding stream. Each simulation is repeated 20 times for all the schedulers, and we analyze the average values.

To simulate different path heterogeneity, we fix the parameters of one path and change the other. We refer to [6] to set the range of bandwidth from 1Mbps to 100Mbps and RTT from 1ms to 400ms.

1) Bandwidth heterogeneity: We set the bandwidth of path #1 to be 1Mbps and the RTT of both paths to be 1ms. We vary the bandwidth of path #2 from 1Mbps to 100Mbps to increase the bandwidth heterogeneity of the two paths.

Fig. 5(a) and (b) illustrate the completion time of time-critical stream A and non-critical stream B under paths with bandwidth heterogeneity, respectively. PStream outperforms the other schedulers in general, especially for the completion time reduction of the time-critical stream A. In LRF, packets are scheduled without considering bandwidth heterogeneity of the paths, so the packets can be distributed over the two paths under suboptimal scheduling decisions. This is even more detrimental to the small streams (e.g. stream A), because suboptimal scheduling decisions will greatly affect the completion time of small streams and cause fluctuation in the repeated experiments due to path heterogeneity. SA-ECF improves LRF by scheduling packets on the path with the lowest estimated completion time. However, SA-ECF fills the congestion window of the better path (i.e. path #2) and idles path #1 for a period of time, and then it transmits packets on the slow path. The burst sending of packets from the fast path requires large buffers and leads to more severe packet out-of-order at the receiver, deteriorating the completion time of the streams. To solve this, PStream allocates paths for the streams, and the scheduled paths can be utilized concurrently by the stream from the beginning of transmission, thus the completion time is reduced. The advantage of PStream on stream B becomes even more evident as the increase of bandwidth difference can aggravate the impact of burst sending.

2) RTT heterogeneity: We set the bandwidth of both paths to be 1Mbps and the RTT of path #1 to be 1ms. Then we vary the RTT of path #2 from 1ms to 400ms.

Fig. 5(c) and (d) depict the completion time of stream A and B under paths with RTT heterogeneity, respectively. For the three schedulers, the performance difference on stream B is not as obvious as that on stream A. This is because the scheduling decision under paths with RTT heterogeneity plays a more important role in determining the completion time of a smaller stream. For stream A, LRF falls short for scheduling packets without considering stream completion time and stream priorities. Comparing to SA-ECF, PStream has an obvious advantage in reducing the completion time of time-critical stream A. This advantage is gradually weakening when SA-ECF uses the worse path at a reduced chance for the increasing of the RTT difference. For stream B, the performance differences are not evident, especially for PStream and SA-ECF. This is because the bandwidth difference of the paths is small, and thus the chance of burst sending by SA-ECF on the fast path is reduced.

TABLE II
PATH PARAMETERS FOR PAGE-LEVEL EVALUATION

Heterogeneity	**# of** `Path(Bandwidth, RTT, PLR)`	
	1	**2**
Low	25Mbps, 10ms, 0	25Mbps, 10ms, 0
Low and Lossy	25Mbps, 10ms, 2%	25Mbps, 10ms, 2%
High	10Mbps, 10ms, 0	40Mbps, 50ms, 0
High and Lossy	10Mbps, 10ms, 2%	40Mbps, 50ms, 2%

3) Bandwidth and RTT heterogeneity: First, we set up two paths with each having its advantage. We set up equivalent initial values of path #1 and path #2, i.e., bandwidth to be 1Mbps and RTT to be 1ms. Then we increase the bandwidth and RTT values of path #2 at the same time.

Fig. 6(a) and (b) show the respective results of stream A and B under paths with each having its own advantage – one with lower RTT and the other with higher bandwidth. The completion time of both streams are reduced evidently in PStream with path heterogeneity. This is due to the mechanism of PStream allows different streams to be scheduled to paths according to their resource preferences. The time-critical small stream A favors the path with lower RTT, while the non-critical big stream B favors the path with higher bandwidth. As a result, the resource contention of the streams on one path is reduced. Under this type of path heterogeneity, SA-ECF loses its advantage for not considering the matching of stream types and path characteristics, thus causing unnecessary inter-stream blocking when a stream competes for its unfavorable path.

Second, we set up two paths where one has both advantages over the other. Path #1 and path #2 are set initially with bandwidth to be 1Mbps and RTT to be 400ms. Then we increase the bandwidth and decrease the RTT values of path #2 simultaneously.

Fig. 6(c) and (d) demonstrate the completion time of stream A and B respectively under paths with one having both advantages over the other. In this scenario, PStream has evident advantages over LRF and slight advantages over SA-ECF. The performance differences are not as evident as in other scenarios. The reason is that the scheduling decisions are similar among the three schedulers because of the overall advantage of path #2 over path #1.

D. Page-level Evaluation

We also evaluate the performance of the three schedulers on web page loading. We deploy websites [26] on the server side, and then we fetch the websites using Chrome as the client [5]. The client is connected to the server through heterogeneous paths with different heterogeneity and packet loss rate (PLR) simulated by Mininet, which is listed in Table II. The experiment is repeated 20 times to get the average results.

Traffic. We choose two famous websites, Bing and Youtube [26], to perform the experiment. The two web pages are of different types. Bing is a simple web page including two kinds of web objects, namely HTML and image, each of which has only one object. Youtube is a complex web page with four kinds of web objects, including one HTML, two CSS, two JavaScript, and twenty image objects.

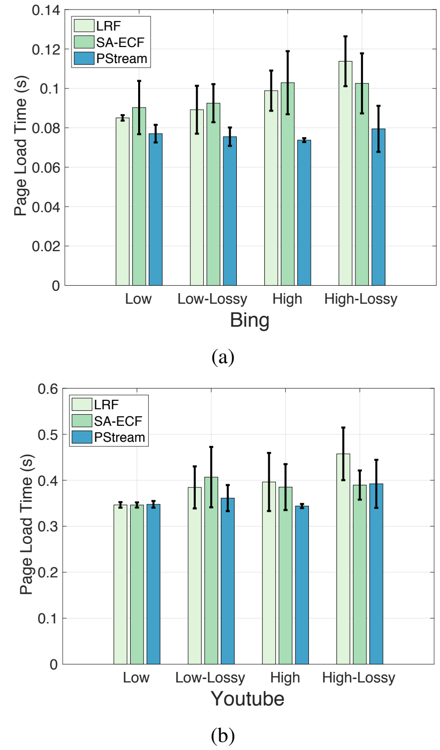


Fig. 7. Average page load time of the three schedulers under different heterogeneity. Each repeated simulation carries a standard deviation value.

Fig. 7 shows the results of Page Load Time (PLT) for Bing and Youtube under different scenarios. In general, PStream performs the best among the schedulers. We observe that the cases with packet losses experience more fluctuation in the repeated simulation. It is because the retransmissions caused by packet losses will increase the uncertainty in the transmission process. When the path heterogeneity gets higher, the PLT results of LRF and SA-ECF increase in most cases. In high heterogeneity, the out-of-order arrival will increase due to the performance difference of paths. This is especially evident in loading Bing, proving that simple web pages are more susceptible to path heterogeneity. In PStream, the mechanism of path allocation can alleviate out-of-order caused by burst transmission on multipath. Meanwhile, the recognition of stream feature in path scheduling reduces contention among streams in heterogeneous paths. Therefore, PStream is more resilient to different degrees of path heterogeneity. In the scenario of high path heterogeneity when loading Bing, PStream can reduce up to 25.4% of PLT. Comparing to Bing, Youtube has a larger amount of objects and the benefits of PStream are less obvious.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a priority-based stream scheduling mechanism, PStream, for MPQUIC. PStream provides stream prioritization to prevent time-critical streams from being blocked by non-critical streams. Streams are scheduled to utilize heterogeneous paths following their application demands. Implementation and experiments proved the effectiveness of PStream, showing that it outperforms the latest scheduler SA-ECF and the original scheduler LRF of MPQUIC in reducing web latency under different path heterogeneity. As to future work, we are interested in extending PStream when facing varying network conditions. Meanwhile, further evaluations of PStream can be explored with different browsers and applications in addition to web browsing – applications such as file download [6] or video streaming [27].

REFERENCES

[1] Tobias Flach et al. Reducing web latency: the virtue of gentle aggression. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*, pages 159–170, 2013.

[2] Matteo Varvello et al. Is the web HTTP/2 yet? In *Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings*, pages 218–232, 2016.

[3] Adam Langley et al. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, pages 183–196, 2017.

[4] Hypertext transfer protocol version 2 (http/2). https://tools.ietf.org/html/rfc7540.

[5] Maarten Wijnants, Robin Marx, Peter Quax, and Wim Lamotte. HTTP/2 prioritization and its impact on web performance. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1755–1764, 2018.

[6] Quentin De Coninck and Olivier Bonaventure. Multipath QUIC: design and evaluation. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2017, Incheon, Republic of Korea, December 12 - 15, 2017*, pages 160–166, 2017.

[7] Tobias Viernickel, Alexander Frömmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. Multipath QUIC: A deployable multipath transport protocol. In *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, pages 1–7, 2018.

[8] Hang Shi et al. STMS: improving MPTCP throughput under heterogeneous networks. In *2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018.*, pages 719–730, 2018.

[9] Quic: A udp-based multiplexed and secure transport, draft-ietf-quic-transport-13. https://tools.ietf.org/html/draft-ietf-quic-transport-13.

[10] Patrick Thiran and Walter Willinger, editors. *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011.* ACM, 2011.

[11] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath TCP. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, pages 399–412, 2012.

[12] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. TCP extensions for multipath operation with multiple addresses. *RFC*, 6824:1–64, 2013.

[13] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, 2001.

[14] Meng Wang, Chee Wei Tan, Weiyu Xu, and Ao Tang. Cost of not splitting in routing: Characterization and estimation. *IEEE/ACM Trans. Netw.*, 19(6):1849–1859, 2011.

[15] Junaid Qadir, Anwaar Ali, Kok-Lim Alvin Yau, Arjuna Sathiaseelan, and Jon Crowcroft. Exploiting the power of multiplicity: A holistic survey of network-layer multipath. *IEEE Commun. Surv. Tutorials*, 17(4):2176–2213, 2015.

[16] Peter B. Key, Laurent Massoulié, and Donald F. Towsley. Path selection and multipath congestion control. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA*, pages 143–151. IEEE, 2007.

[17] Alexander Rabitsch, Per Hurtig, and Anna Brunström. A stream-aware multipath QUIC scheduler for heterogeneous paths: Paper # xxx, XXX pages. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, EPIQ@CoNEXT 2018, Heraklion, Greece, December 4, 2018*, pages 29–35, 2018.

[18] Xiang Shi, Lin Wang, Fa Zhang, and Zhiyong Liu. Fstream: Flexible stream scheduling and prioritizing in multipath-quic. In *25th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2019, Tianjin, China, December 4-6, 2019*, pages 921–924. IEEE, 2019.

[19] Jing Wang, Yunfeng Gao, and Chenren Xu. A multipath QUIC scheduler for mobile HTTP/2. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking, APNet 2019, Beijing, China, August 17-18, 2019.*, pages 43–49, 2019.

[20] Rasmus Suhr Mogensen. Reliability enhancement for lte using mpquic in a mixed traffic scenario. In *Technical Report, Aalborg University, Aalborg, Denmark, June 7, 2018*, 2018.

[21] Van Jacobson. Congestion avoidance and control. In *SIGCOMM '88, Proceedings of the ACM Symposium on Communications Architectures and Protocols, Stanford, CA, USA, August 16-18, 1988*, pages 314–329, 1988.

[22] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. TCP westwood: End-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8(5):467–479, 2002.

[23] PStream. https://github.com/Xiang-Shi/PStream.

[24] Neal Cardwell et al. BBR: congestion-based congestion control. *ACM Queue*, 14(5):20–53, 2016.

[25] Nikhil Handigol et al. Reproducible network experiments using container-based emulation. In *Conference on emerging Networking Experiments and Technologies, CoNEXT '12, Nice, France - December 10 - 13, 2012*, pages 253–264, 2012.

[26] Recorded dependency graph and web objects of famous web pages. http://wprof.cs.washington.edu/spdy/tool/.

[27] Jiyan Wu, Chau Yuen, Bo Cheng, Ming Wang, and Junliang Chen. Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks. *IEEE Trans. Mob. Comput.*, 15(9):2345–2361, 2016.