Deadline-aware Multipath Transmission for Streaming Blocks

Xutong Zuo*, Yong Cui*§, Xin Wang†, Jiayu Yang‡
*Tsinghua University, China
†State University of New York at Stony Brook, USA
‡Beijing University of Posts and Telecommunications, China

***Abstract*—Interactive applications have deadline requirements, e.g. video conferencing and online gaming. Compared with a single path, which may be less stable or bandwidth insufficient, using multiple network paths simultaneously (e.g., WiFi and cellular network) can leverage the ability of multiple paths to service for the deadline. However, existing multipath schedulers usually ignore the deadline and the influence from subsequent blocks to the current scheduling decision when multiple blocks exist at the sender. In this paper, we propose DAMS, a Deadline-Aware Multipath Scheduler aiming to deliver more blocks with heterogeneous attributes before their deadlines. DAMS carefully schedules the sending order of blocks and balances its allocation on multiple paths to reduce the waste of bandwidth resources with the consideration of the block's deadline. We implement DAMS with the inspiration of MPQUIC in user space. The extensive experimental results show that DAMS brings 41%-63% performance improvement on average compared with existing multipath solutions.**

I. Introduction

Applications often put many requirements on the services from network transmissions. Among the requests, end-to-end delay is most concerned by interactive applications, such as video conferencing and online gaming, which usually expect the data to arrive within a certain time limit, i.e., a deadline. For instance, in a video conference, the end-to-end delay is supposed to be about 100ms or even lower to provide users with interactive experience [1]. In recent years, teleconference has become an essential tool to support normal business operations, and there is also a large increase of online gaming. It is critical to well support low-delay interactive applications.

The data from streaming applications are often transmitted in blocks. Data block is defined as the minimal unit of data by applications. For example, frames in video streaming and messages in online gaming can be treated as blocks. For block-based data transmission, the order in which data arrives within a block is unimportant, as long as all data of a certain block reach the receiver before the block's deadline. The blocks that miss the deadline will greatly affect the user's QoE, and may become useless when new data arrive.

Previous work of deadline-aware transmissions on a single path [2], [3] attempts to deliver data blocks before application-specified deadlines. However, a single path can not always provide stable network connections or sufficient throughput [4], [5]. Compared with a single path, multipath transmissions have the advantage of providing seamless handover and larger aggregated bandwidth, which show their potential in dealing with deadline-aware transmissions. Besides, many applications now support multipath solutions [6], [7]. For multipath transmission, a scheduler is often used to determine which path the data should be transmitted along, and is the key component that impacts the performance. Various multipath schedulers are proposed for different scenarios [8]–[11] and optimization objectives [12]–[15].

However, existing multipath schedulers can not well handle deadline requirements for streaming blocks as they often ignore deadlines and the interactions among blocks (§ II). MinRTT [16], Round-robin (RR) and ECF [17] are not aware of the block boundary and deadline requirements. If the one-way delay of the large RTT path is greater than the deadline of a block, the block will miss the deadline. Although block boundary is considered in DEMS [18], it only tries to minimize the completion time of a single block, without considering the interaction of blocks. DEMS achieves simultaneous subflow completion from the receiver view, however, its bandwidth allocation increases the bandwidth wastage. As a result, some blocks meet their deadlines by sacrificing other blocks. MP-DASH [4] tries to ensure a single block to meet the deadline, but is difficult to handle the situation with concurrent blocks. Existing algorithms proposed for concurrent streaming on multipath are either based on SCTP [19] which is difficult to deploy [20], or intend to optimize other metrics, such as the average stream completion time (SRPT-ECF) [21], instead of the deadline.

This paper aims to enable deadline-aware multipath delivery for streaming blocks. Achieving this goal faces a number of challenges. First, deadline-aware schedulers need to arrange the sending order of blocks which may have heterogeneous and contradictory attributes, e.g. deadline, priority and size. Second, subsequent unknown blocks affect the current multipath bandwidth allocation. Third, contradiction exists between the precise deadline and the inaccurate network information. The key contribution of this paper is the design, implementation and evaluation of **D**eadline-**A**ware **M**ultipath **S**cheduler (DAMS) for transmission of streaming blocks. DAMS addresses the above challenges as follows:

- To deliver streaming blocks within application-specific

§Yong Cui is the corresponding author.

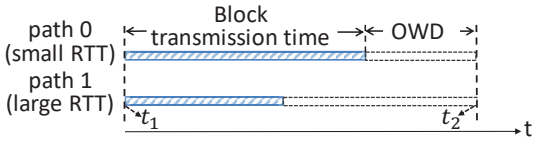


Fig. 1. Simultaneous subflow completion of chunk i which achieves the minimum completion time. Block transmission time is calculated as the ratio of block size to bandwidth. With no preemption, the completion time of a block includes two parts: block transmission time and one-way delay (OWD). Block i starts sending at t_1 and completes at t_2.

deadlines in the multipath scenario, DAMS considers the deadline as an important metric to handle the sending order of blocks with heterogeneous attributes. During the scheduling, DAMS detects the blocks' deadline satisfaction with network capacity instead of by estimating the transmission time, which decouples the specific multipath allocation strategies from the decision of block sending order (§ III-A – III-B).

- Future blocks and network changes cause the transmission of the current block to be preempted which lead to bandwidth waste. The multipath allocation strategy of DAMS is to complete the sending simultaneously on multiple paths for each block to reduce the impact of preemption and achieve high performance (§ III-C–III-D).

We implement DAMS in user space based on QUIC [22] for simplicity with the inspirations of DTP [2] and MPQUIC [13] (§ IV). We conducted extensive evaluations in stable and dynamic networks. Our evaluation results show that DAMS achieves the best QoE metrics compared with other multipath schedulers. Additionally, our results demonstrate that DAMS can increase the completion rate of high priority blocks up to 48% and the completion rate of all blocks up to 71% for different applications under various network conditions (§ V).

II. Motivation

For many delay-sensitive applications, data are transmitted in the unit of block with its *deadline*. The data blocks may have different impacts on the user-perceived QoE. The blocks that show significant influence on QoE are considered to be important with a high *priority*, such as I frames and tiles that the user's eyes focus on in a 360° video [23]. Multiple blocks may be sent simultaneously. A natural question is how to leverage multipath resources and blocks' attributes so that more blocks can meet their deadlines. Before proposing a solution, we investigate the reasons for the performance degradation caused by previous multipath schedulers.

For deadline-aware multipath scheduling, one major cause of performance degradation is that existing scheduling schemes waste the bandwidth. For deadline-oriented transmission, if data arrive after the deadline, the used bandwidth is also considered to be wasted. We find that previous multipath schedulers are unsuitable for deadline-aware transmissions, because they do not consider multiple blocks in the deadline scenario, leading to bandwidth waste. Next, we analyze the reasons with examples.

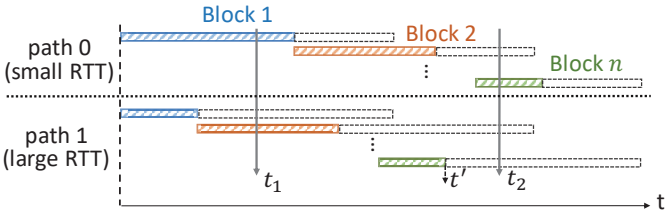


Fig. 2. The timeline of multiple blocks sent in the arranged order. The multipath allocation strategy is simultaneously subflow completion (A multi-block version of Figure 1). Above and below the dotted line are bandwidth allocations for the first and second paths, respectively. Multiple blocks are shown with different colors.

Existing data sending order does not take into account the heterogeneous attributes of multiple blocks. When multiple blocks with different attributes (e.g., deadline, priority, and size) exist at the sender, FIFO sending order of blocks is inadequate. An intuitive example is that, if a block misses its deadline before the transmission completes, the remaining data of that block becomes useless even if transmitted to the receiver. In this circumstance, it is a better choice to delay or cancel the sending of this block and make way for subsequent data transmission to avoid wasting the bandwidth.

What's more, simply considering these attributes may still cause unexpected results. For example, if only priority is considered and a high priority block (chunk A) far away from its deadline is sent first, a low priority block (chunk B) close to its deadline may miss the deadline and get canceled. Only considering the block size leads to the same problem. Considering the deadline is not enough either, because whether a block can be completed is related to not only the deadline, but also the block transmission time, propagation time, etc. Choosing a block whose deadline is close but may not be completed before the deadline is not reasonable.

These examples and analysis indicate that simply considering attributes of blocks is not suitable for deadline-aware transmission of streaming blocks, let alone FIFO which ignores the attributes. Besides, due to the existence of the deadline, the time that each block can use the bandwidth is constrained, and bandwidth allocation needs to be well designed. Consequently, a better block sending order which considers the deadline and other heterogeneous attributes of blocks is needed, with the aim of increasing the block completion rate and reducing the waste of bandwidth resources.

Mismatch exists between current multipath allocation strategies for a single block and the deadline target. For a single block, the minimum block completion time is attained by *simultaneous subflow completion* multipath strategy [18], [21] (shown in Figure 1). Even so, we find that when faced with transmission streaming blocks, such a scheme would cause frequent preemption, which may lead to potential bandwidth waste.

With this strategy, the multipath allocation of each block aims at *minimizing the block completion time*. When multiple blocks exist at the sender (e.g. n blocks), the bandwidth occupancy of these n blocks is shown in Figure 2. For the first

$n-1$ blocks, following the simultaneous subflow completion strategy, two paths often transmit different blocks at the same time. If there is a preemption (e.g., at the time t_1), more than one block may be affected, and multiple preempted blocks may all miss their deadlines to cause a big waste of bandwidth. What's more, the larger the difference of the one-way delay between the two paths, the greater the difference of the block transmission time on different paths for each block. As a result, the bandwidth waste mentioned above will occur more frequently.

As for the n_{th} block, that is, the last block scheduled to be sent in the current buffer, simultaneous subflow completion leads to the potential bandwidth waste on the slow path. As shown in Figure 2, after t', block n only occupies the fast path, and the slow path is idle without new blocks coming. However, if the one-way delay of the slow path is less than the deadline of block n, the slow path should have been used to speed up the existing transmissions rather than being wasted (from t'). It would also reduce the probability for the block n to be preempted, as in the case that a new block arrives at t_2. The impact and performance degradation caused by preemption is shown with the evaluation results in Section V-C.

As a result, simultaneous subflow completion increases the possibility of bandwidth resource competition due to unreasonable bandwidth allocation schemes. Therefore, to improve the completion rate of blocks and better support deadline-aware transmissions, it calls for the design of an efficient multipath scheduler to reduce the bandwidth waste and competition.

III. Deadline-aware multipath scheduler

We propose DAMS, a deadline-aware multipath scheduler aiming to deliver more blocks with heterogeneous attributes before their deadlines. The key design decisions of DAMS include the following: (1) DAMS is aware of the blocks' deadlines (III-A); (2) To handle heterogeneous attributes of blocks, DAMS strategically decides the sending order of blocks with a variant of Earliest Deadline First (EDF) algorithm (III-B); (3) To deal with unknown application data pattern, for practical use, DAMS makes the online adaptation for multipath bandwidth allocation (III-C); (4) To deal with unknown network status, Robust DAMS (III-D) is designed where a safeguard is set to adjust the deadlines of blocks for scheduling. Next, we elaborate them in this section.

A. Overview for deadline-aware block transmission

DAMS aims to achieve block-based transmission with deadline. Each block has attributes such as deadline, priority and size. For blocks, the deadline determines how long a block can occupy bandwidth, the priority indicates the impact of a block on the user QoE, and the size determines the transmission time of a block. What DAMS concerns is the completion time of blocks, which affects the user experience.

Section II shows that FIFO is not the best sending order of blocks for deadline-aware transmission. Other schemes that consider only one of the attributes cannot provide a reasonable sending order. Under this circumstance, DAMS aims to strategically decide the sending order of blocks with different properties taken into account.

When faced with the deadline-aware scheduling, a straightforward solution is the Earliest Deadline First algorithm (EDF) [24] used for processor scheduling in a real-time system. However, most research focuses on the problem of homogeneous multi-resources [25]–[27] or uniform heterogeneous multi-resources [28]–[30]. It is different from our scenario, and the solution cannot be directly applied. Specifically, vanilla EDF algorithms are not suitable for the following reasons:

- Different from heterogeneous processor resources, multipath resources are not only different in bandwidth but also latency which is an important factor affecting the scheduling.
- In most cases, network is overloaded with limited resources, and no schedule can meet the deadlines of all blocks. At this point, we focus on how to meet the deadlines of more blocks. However, most studies on this aspect in real-time scheduling cannot be directly applied.

To solve the above problems, we first propose an offline EDF-based algorithm that considers heterogeneous attributes to give a block sending order when the global information is known, and then make adaptations for practice use. The workflow of DAMS is shown in Figure 3. DAMS takes block attributes and network status as input, and decides the block sending order and the distribution of the block data on multipath with the consideration of deadline. Blocks' attributes are notified to the transport layer which is implemented as Section IV shows. Network status includes bandwidth and RTT. We estimate the bandwidth by calculating the ratio of congestion window and RTT. One-way delay is estimated with $RTT/2$. However, this estimation may contain flaws, as the delay of an Internet path can be asymmetric [31], [32] and the lagged nature of RTT still impacts the perception of DAMS for network delay, thus the scheduling to make.

B. Offline scheduling

In this section, we propose the offline DAMS when the information of network condition and all blocks are given. The scheduling is still challenging even with these information, as multiple heterogeneous attributes of blocks which conflict each other and should be handled. Next, we propose an EDF-based scheduling algorithm and show the design details as follows.

DAMS performs *order arrangement* (in Figure 3) and sorts all blocks according to their deadlines. If there is a resource allocation scheme so that all blocks in the sender buffer can meet the deadline, the optimal solution can be obtained by EDF. However, network resources are usually insufficient and not all blocks can meet their deadlines. In this circumstance, DAMS performs *selective preservation* which selectively delays or cancels the sending of some blocks that are sorted according to the deadline. Given a block of size s, its percentage of unsent data is denoted as r. If the block arrives before its deadline, the credit c is obtained. The credit c is positively correlated with the priority p. DAMS selectively preserves some blocks. Specifically, DAMS performs a block-by-block comparison of

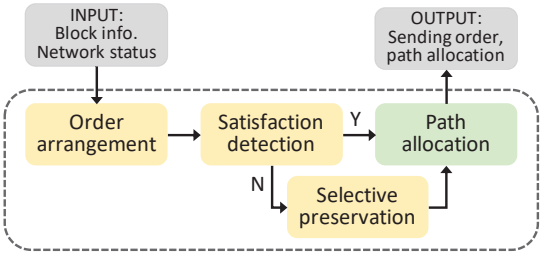


Fig. 3. Workflow of DAMS algorithm.

all blocks. If the deadline of the comparison blocks can be met with current network conditions, the comparison blocks will not be delayed or cancelled. If not, delaying the sending of blocks with the smallest normalized credit $d = \frac{c}{s*r}$, where a larger d means a higher credit for transmitting the same amount of data. Once time exceeds the deadline of a block, it will be cancelled and will no longer participate in scheduling afterwards. This helps reduce the bandwidth waste caused by sending useless data. Meanwhile, the queuing delay of other blocks can be reduced.

If all blocks have the same priority, DAMS selects the smallest block when scheduling and achieves the optimal solution. However, when considering different priority, tradeoff exists between the priority and block size, and the optimal solution can only be found by traversing all situations. In this circumstance, DAMS performs selective preservation according to the normalized credit d.

In the offline phase, the multipath scheduling scheme only needs to satisfy that the blocks which are not delayed or cancelled are completed before their deadlines. A natural question arises: how to detect that the deadline of a block can be met. A direct method is to estimate the completion time of the block and compare it with its deadline. However, the completion time of the block is related to the multipath allocation of that block, and is affected by the scheduling of other blocks as well. The existence of the coupling relationship makes the problem complicated. Notably, we find that the available network capacity before the deadline for each block can be used to detect whether the block can meet the deadline or not. If the deadline can be met, any specific allocation across multipath which satisfies the deadline can be used. Consequently, detecting the deadline satisfaction is decoupled from multipath allocation and is performed according to the network capacity.

Specifically, the detailed calculation of network capacity that a block can use is shown as follows. The attributes of heterogeneous paths need to be considered, including bandwidth and RTT. The maximum network capacity that block i can use on path j is:

$$C_i = \sum_j (D_i - A_i - RTT_j/2) \times BW_j \tag{1}$$

where D_i and A_i represent the deadline and arrival time of block i respectively. Besides, the RTT_j and BW_j represent the RTT and bandwidth of path j. However, due to the insufficient network bandwidth, block i may be queued at the sender and waits for other blocks to be sent. As a consequence, the available network capacity for block i is reduced to $C_i - C_{used}$ where C_{used} represents the capacity that is used by other blocks sent before block i.

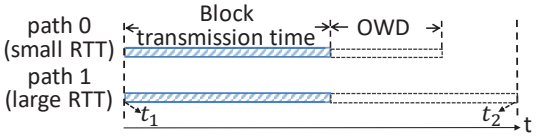


Fig. 4. Finish sending simultaneously. Transmission timeline of a block in DAMS. The block starts sending at t_1 and completes at t_2.

C. Online adaptation for multipath bandwidth allocation

When online, unlike the offline phase, only the information of the blocks in the sending buffer and the current network status can be known. The challenge arises when considering multipath scheduling for a block, as the impact of unknown subsequent blocks on the current decision should be taken into account. New blocks and network condition changes may lead to preemption of current blocks. Therefore, multipath allocation should be carefully designed for online scheduling.

From the previous block-based solution DEMS [18], we know that simultaneous subflow completion generates the minimum completion time of a block. In this case, the block has the highest slack and presents high tolerance for network jitter. However, as Figure 2 shows, in the circumstance of simultaneous subflow completion, multiple paths usually transmit different blocks at the same time. Consequently, when a new block comes, more than one block is preempted, resulting in an increase in bandwidth waste. The greater the delay difference between different paths, the worse the situation.

In order to reduce the bandwidth waste, when each block is sent, the *path allocation* strategy of DAMS is to finish the sending simultaneously on multiple paths. By doing so, the same block is ideally sent on multiple paths at the same time, so at most one block will be affected when a preemption occurs. The transmission timeline of a block is shown in Figure 4. If the capacity is enough for all blocks which are not delayed or cancelled, there exist multiple allocation schemes that can meet the deadline of blocks. Among them, finishing the *sending* on multiple paths simultaneously for each block achieves the optimal multipath allocation strategy with the minimum bandwidth waste.

Finishing the sending simultaneously makes the completion time of the block on multiple subflows different, and the path with a large one-way delay completes transmission later. If even the smallest completion time of multiple paths exceeds the deadline, the block is expected to miss its deadline. A special case may occur that the largest completion time of multiple paths exceeds the deadline while the smallest one not. If this happens, we adjust the allocation by moving the part of the data beyond the deadline to the path with smaller

one-way delay. Furthermore, compared with the simultaneous subflow completion, this allocation method has a lower ability to handle the network fluctuation on a large RTT path. So in a dynamic network, we take an extra operation to deal with the network fluctuation (Section III-D).

The overall operating logic and complexity analysis of DAMS are as follows. When a new block comes, DAMS inserts it into the queue according to its deadline, and then executes the above *selective preservation* procedure. For each block, DAMS tries to finish its sending simultaneously on multiple paths. The algorithm complexity is $O(k)$, where k is the number of blocks whose deadlines are behind the new block in the original buffer. When a new block comes or network condition changes, the blocks in the sending buffer are rescheduled. To prevent the scheduling overhead from increasing indefinitely as the number of blocks increases, in practice use, only the front part of the block can participate in scheduling to achieve a trade-off between algorithm performance and overhead. Notably, the design of DAMS is not targeted specifically for two paths, thus DAMS can be applied directly to scenarios with more than two paths, which shows its flexibility and scalability.

D. Deadline adjustment for network dynamics

The DAMS scheduling algorithm shown above considers the bandwidth and RTT provided by congestion control module as accurate network conditions. However, in a dynamic network, it is impossible to provide accurate predictions to the scheduling algorithm. When the network condition changes, the sent data which scheduled with previous network conditions may miss their deadline, resulting in the bandwidth wastage.

To handle varying network, we enhance the robustness of DAMS with the adjusted deadline as input. First, RTT is estimated at the sender with exponential weighted moving average (EWMA). Then, in order to avoid the situation that blocks that could be completed before the deadline miss it due to the smaller throughput or larger RTT afterwards, we add a safeguard for the deadline and get $\overline{Deadline}$:

$$\overline{Deadline} = Deadline - \alpha * std\_dev/2 \tag{2}$$

where *std_dev* varies with time, representing the standard deviation of RTT collected in the past period of time. $std\_dev/2$ is used as the variation of one-way delay. We multiply *std_dev* by a coefficient α to adjust for the conservative level and $\overline{Deadline}$ is used as the input of scheduling. By adding the safeguard, Robust DAMS becomes more conservative because it use a smaller deadline and the blocks near the deadline will be delayed or cancelled. Doing so increases the probability that sent blocks complete before the deadline, and thereby reduces the waste of bandwidth resources caused by sending useless overdue blocks. When the network bandwidth and RTT stay stable, std_dev is small, then the performance of the robust DAMS algorithm is similar to that of the DAMS algorithm mentioned above. However if the network fluctuates significantly, the robust DAMS algorithm improves the block completion rate before the deadline.

IV. Implementation

The implementation of DAMS puts requirements for the underlying protocol stack. First, it should be able to mark the block boundary and be aware of the deadline corresponding to each block in the data stream. Second, it is supposed to support unreliable or partial reliable transmission to enable block cancellation in DAMS. These requirements make it a great effort to implement DAMS in MPTCP, though it is possible, and we leave it for future work. Owing to many useful building blocks in QUIC, DAMS is implemented based on MPQUIC without much effort. Next, we show our primary extensions of QUIC and MPQUIC.

To be able to cancel blocks that miss the deadline, we map a block to a stream of QUIC, so we take advantage of the existing stream cancellation process of QUIC to cancel blocks without influencing transmission of other blocks. QUIC stream ID is encoded using variable-length integers of which the max value is $2^{62}-1$, so it is enough to represent all blocks during one session. Once DAMS decides to cancel a block, it calls the function *cancel_block* implemented in our system, and the sender will remove it from the sender buffer. Meanwhile, a RESET_STREAM of QUIC is sent to inform the receiver that the block is canceled. Mapping a block to a stream of QUIC also enables prioritizing some blocks over others.

In MPQUIC, data retransmissions do not have to follow the original path. This property allows us to schedule retransmitted data like other blocks without need of special treatment. In our implementation, the retransmitted data are treated as a part of the original block and participate in multipath scheduling. If the retransmitted data miss the block deadline, we can also cancel the sending like other blocks. Consequently, not all lost packets will be retransmitted.

For DAMS, the scheduling performed on the packet level means high overhead. Next, we present the conditions for triggering the scheduler on the block level to lower the scheduling overhead. First, if the network conditions of the two paths do not change, the sending order of blocks and the allocation remain the same. So the scheduling is triggered when this module perceives network changes. Second, when new blocks come to the sender buffer, preemption may occur so scheduling should be triggered.

We use the basic implementation on Cloudflare's quiche of the version 23 of QUIC draft [33]. We extend it with inspirations from DTP [2], [3] and MPQUIC [13]. Our implementation is written in Rust and the example application is written in C. The overall implementation is lightweight. With DAMS, we achieve deadline-aware transmission at the transport layer, facilitating the application developer by providing only an additional deadline parameter. Application proxy can be used to avoid modifying the applications. Besides, the deadline setting is configurable and application-related.

V. Evaluation

In this section, we conduct experiments on two Linux machines running Ubuntu 20.04. One serves as the sender with scheduling algorithm deployed on it and the other serves as the

receiver. They are connected with an OpenWrt router where Linux tc [34] runs to provide different network conditions. To create two transmissions, two network interface cards are installed on each machine.

A. Experiment Setup

Application traces. We evaluate DAMS with two application traces on top of our system.

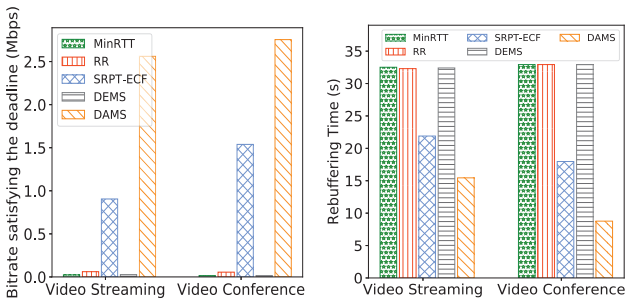
- Live video streaming (Advanced Video Coding, AVC). We run our video streaming emulator with the "AirShow" video from the LIVE video set [35]. We encode the trace to the bitrate of 4.3Mbps and parse them to the frame level with FFmpeg to simulate low-latency live streams. Each frame is treated as a block. The parsed video has three types of frames: I, P, and B with high, medium, and low priority respectively. The frame rate is 30FPS.
- Video conference (Scalable Video Coding, SVC). We use the "Chimera1102347" video from the LIVE video set [35]. Video conference applications adopt scalable video coding, where videos are encoded into base layer and enhancement layers, which are treated as blocks with different priorities. The highest bitrate of SVC video is 4.3Mbps. The original video is 30FPS.

Comparison algorithms. In our evaluation, we compare DAMS's performance to that of the following five comparision algorithms.

- MinRTT [12]: sends packets through the available paths with the smallest RTT.
- RR: sends packets in a round-robin fashion.
- SRPT-ECF [21]: selects the smallest stream and for each stream it tries to minimize overall completion time.
- DEMS [18]: block-based multipath scheduler. For each block, it tries to achieve simultaneous subflow completion from the receiver's view.
- DAMS-C: block-based multipath scheduler, a variant of DAMS, with our designed block sending order. For multipath scheduling, it tries to achieve simultaneous subflow completion from the receiver-side view.

We present the overall performance of DAMS comparing with existing comparison algorithms of the two applications motioned above in Section V-B. In addition, to provide a deeper understanding of DAMS, we describe microbenchmarks where DAMS are compared with two algorithms: DEMS (the representative of existing block-based algorithms) and DAMS-C (the variant of DAMS). For fair comparision, we implemented all the comparision algorithms in our system, so that they share the same underlying mechanism.

Performance metrics. We present the evaluation using metrics of application layer and transport layer. We evaluate DAMS and other comparison algorithms with the same source video and network trace, and present the *block completion rate of all blocks and high-priority blocks* as metrics of the transport layer. Besides, we calculate the *average bitrate* and *rebuffering time* at the application layer. Notably, in the deadline scenario, only blocks that arrive before the deadline are valid and counted. Specifically, the average bitrate is calculated as $data\_volume/video\_len$ where $data\_volume$ represents the amount of data belonging to the blocks that arrived before the deadline and $video\_len$ represents the length of the video. The rebuffering time is calculated as $frame\_num/frame\_rate$ where $frame\_num$ is the number of frames that missed the deadline. It comes from the fact that every frame missing the deadline introduces $1/frame\_rate$ rebuffering time.



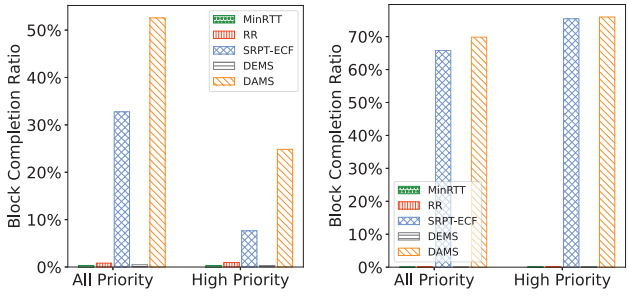
(a) Average Bitrate. (b) Rebuffering Time.

Fig. 5. QoE of applications.

B. Overall performance

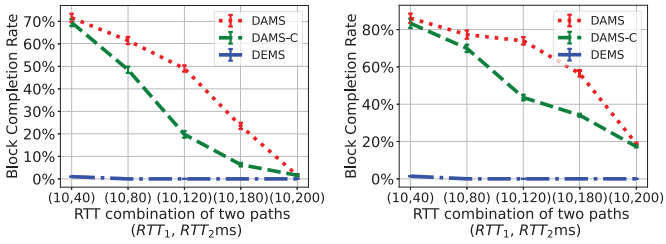
We now present the overall performance of DAMS when comparing with existing multipath algorithms: RR, MinRTT, SRPT-ECF and DEMS in Figure 5 and Figure 6. We use real network traces randomly selected from the HSDPA dataset [36] to simulate dynamic network conditions. The average bandwidth of the two paths are 1.50Mbps and 2.36Mbps respectively. The aggregated bandwidth is lower than the average bitrate of the application trace. The minimum RTT of the two paths are set to 10ms and 40ms respectively. The deadline of blocks is 200ms.

Figure 5 shows the QoE of two applications. In the deadline scenario, DAMS performs the best when comparing with other existing algorithms. DAMS achieves the highest average bitrate which proceeds 2.5Mbps. Besides, DAMS achieves the lowest rebuffering time for both applications which is 48.39% of the best performing comparison algorithm SRPT-ECF. MinRTT, RR and DEMS are not aware of deadline, and the data sent by the sender is FIFO. As a result, with the limited bandwidth, most blocks cannot meet the deadline due to long queuing delay at the sender, which leads to low average bitrate and long rebuffering time. We revisit the question of long queuing delay in Section V-C. SRPT-ECF performs better than the above three algorithms because it prioritizes sending blocks with small size. However, its block preemption only considers the size without considering the priority and the amount of sent data in the current block. This makes SRPT-ECF perform worse in the live video streaming application than in video conference, because the data block size in our video conference trace is smaller than another application. To further explain it, we then give detailed analysis with the data pattern of the two applications and the block completion rate metric which is shown in Figure 6.



(a) Video streaming. (b) Video conference.

Fig. 6. Block completion rate of applications.



(a) Block completion rate of high-priority blocks. (b) Block completion rate of all blocks.

Fig. 7. Overall performance under different RTT combination.

Application data patterns of these two applications are different. Specifically, in AVC, the highest priority block has the largest size (I frame), and in SVC the highest priority block has the smallest size (base layer). As Figure 6 shows, for video conference, the completion rate of high-priority blocks is higher than that of all blocks, while that of video streaming is the opposite. The reason is that, more small and high-priority blocks are completed before deadline in video conference compared with low-priority blocks. Besides, SRPT-ECF achieves comparable performance with DAMS in video conference scenario as DAMS also preserves the blocks with small size and high priority. More results showing the effect of multipath allocation under different networks are presented in Section V-C. For average bitrate, as shown in Figure 5(a), the results is consistent with the trend for the completion rate of all blocks in Figure 6. For rebuffering, since in a video conference, once the base layer reaches the receiver, this frame does not cause rebuffering. Therefore, there is less rebuffering in video conference application than in AVC based video streaming as Figure 5(b) shows.

C. DAMS deep dive

In order to better explore the influence of the multipath allocation strategies on the results, we implement a variant of DAMS and obtain DAMS-C. This scheduler is different from DAMS only in multipath allocation, and it achieves simultaneous subflow completion which is the same as DEMS. Besides, we choose DEMS as a representative of the existing algorithms for comparison because it is a block-based algorithm and the multipath allocation strategy is the same as DAMS-C. Next, we evaluate the three algorithms of DEMS, DAMS, and DAMS-C under stable network and dynamic network conditions with different bandwidth and RTT combinations. For the block-based algorithm, by comparing DEMS with the other two algorithms, we present the role of block sending order in our design. In addition to that, we evaluate the multipath allocation strategy by comparing DAMS and DAMS-C. We use live video streaming trace as an example application to carry out subsequent experiments. The deadline of each block is 200ms.

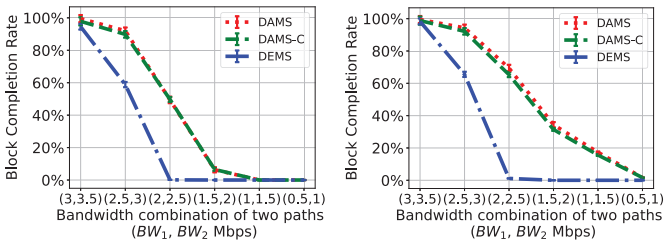
1) Stable Network Conditions: We consider different RTT and bandwidth differences of two paths, and the network settings follow the methodology of controlled experiments in [4] and [18]. We measure the block completion rate of high priority blocks and all blocks. We repeat each experiment 5 times and report the average value.

Different RTT combinations. We fix the first path's RTT to 10ms and vary RTT of the second path from 40ms to 200ms with equal interval 40ms. The bandwidth of the two paths are set to 2Mbps and 3Mbps.

Figure 7 shows that DAMS performs the best. DEMS performs the worst and the aggregate bandwidth of DEMS is limited, so almost no blocks can be completed before the deadline. The limited aggregate bandwidth might be due to the fact that DEMS sends the next block after the previous one finishes sending on both paths instead of one of the paths, which results in a waste of bandwidth on the large RTT path even though it finishes the transmission earlier. Compared with DAMS-C, DAMS achieves up to 29% and 32% higher completion rates for high-priority blocks and all blocks respectively. In our video streaming trace, I frame with the largest size is marked as the high priority blocks. With the long block transmission time, the completion rate of high priority blocks reduces rapidly as the RTT increase (Figure 7(a)). Besides, as the delay difference of the two paths increases, the performance difference between DAMS-C and DAMS first increases and then decreases. The initial increase is because the competition and preemption of the small RTT path increase as the RTT difference becomes larger. If the RTT of the second path is so large that the multipath allocations of DAMS and DAMS-C tend to be consistent, the performance difference becomes smaller. The completion rate of all blocks in Figure 7(b) shows the same trend.

To understand where the performance improvement of different multipath strategy comes from, we compare DAMS with DAMS-C and show the competition and preemption by measuring the number of blocks which have ever sent data but are eventually cancelled. These blocks reflect the degree of competition and preemption, as otherwise a block will be sent over without being preempted. Figure 9 shows that more blocks are cancelled by DAMS-C than DAMS. The arrived data belongs to the cancelled blocks also become useless. The more the cancelled blocks which ever be sent, the more bandwidth is wasted, resulting in lower block completion rate.

Different bandwidth combinations. To understand the



(a) Block completion rate of high-priority blocks.

(b) Block completion rate of all blocks.

Fig. 8. Overall performance under different bandwidth combination.

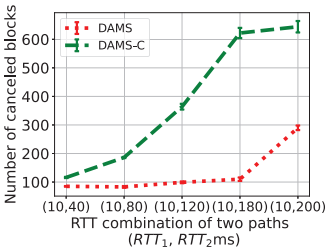


Fig. 9. Number of canceled blocks.

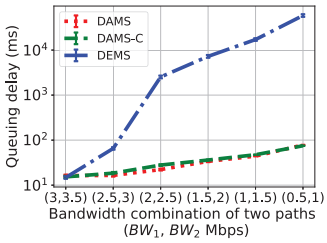
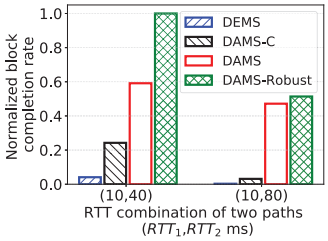
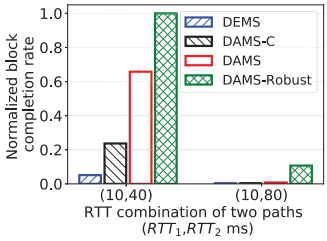


Fig. 10. Queuing delay at the sender on the first path.



(a) Block completion rate of all blocks.



(b) High priority block completion rate.

Fig. 11. Overall performance under dynamic network setting 1.

influence of two paths' bandwidth, we evaluate the algorithms under different bandwidth combination of two paths. We vary the bandwidth of two paths and compare the block completion ratio under 6 different bandwidth combinations. We set the RTT of two paths to 10ms and 20ms separately. The random loss rate to 0.01%.

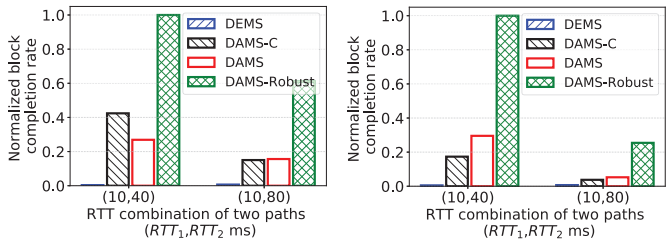
Figure 8(a) shows the completion rate of high priority blocks that arrive before deadline. The Y-axis is the completion rate and the X-axis is the bandwidth combination pair of two paths' bandwidth: (3, 3.5), (2.5, 3), (2, 2.5), (1.5, 2), (1, 1.5), (0.5, 1). The bandwidth of two paths decreases from the left to the right on X-axis, so does the aggregated bandwidth. Figure 8(a) shows that DAMS and DAMS-C which apply the block sending order we designed perform better than the DEMS which applies FIFO when dealing with multiple blocks at the sender. This shows the effectiveness of selective preservation when many blocks exist at the sender with limited aggregate bandwidth. Moreover, the performance decrease of schedulers is different. For DEMS, if the aggregate bandwidth is limited, data are accumulated at the sender so that the queuing delay increases and becomes the culprit for the block not being completed. For DAMS and DAMS-C, the long block transmission time becomes the limiting factor for the improvement of the completion rate. Other bandwidth combinations present similar performance to those with similar aggregated bandwidth shown in Figure 8, so we omit them.

To reveal the performance gain brought by selective preservation, we measure the queuing delay at the sender of blocks that reach the receiver. Due to the existence of preemption, the queuing delay also includes the time preempted by other blocks in addition to the waiting time before starts sending. The queuing delay on the first path is shown in Figure 10 (The result is similar on the second path, so we omit the figure). We can find that the queuing delay of algorithms with selective preservation is lower than DEMS and keeps consistently low. DEMS applies FIFO, so more and more blocks accumulate at the sender due to limited bandwidth and the average queuing delay increases. Selectively delaying or canceling the sending of some blocks reduces the queuing delay and prevent the accumulation of queue.

The completion rate of all blocks that arrive before the deadline in Figure 8(b) shows the same trend as Figure 8(a). When the aggregate bandwidth is sufficient, nearly all blocks can arrive before the deadline. When the aggregate bandwidth is limited, around the point (2Mbps, 2.5Mbps), we find that the performance of DEMS degrades dramatically and nearly no blocks can complete before the deadline. Overall, compared with DEMS, DAMS achieves up to 48% and 71% increase in the completion rates for high-priority blocks and all blocks respectively. As Figure 8 shows, the performance difference between DAMS-C and DAMS is minor as the RTT difference of the two paths is small.

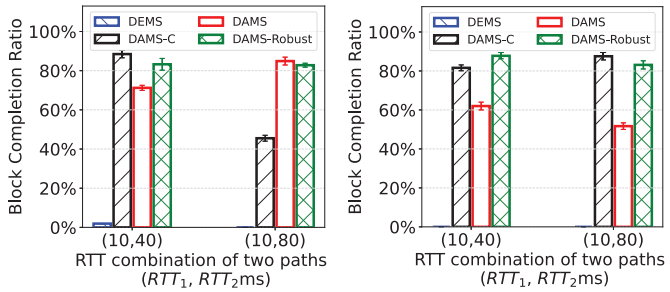
2) Dynamic Network Conditions: The algorithm to be compared includes: DEMS, DAMS-C, DAMS and DAMS-Robust. We set α as 1 empirically in DAMS-Robust and leave the work about the effect of α on performance to the future. We vary the bandwidth of two paths during one test. We evaluate in two dynamic network setting with bandwidth of different average and variance: 1) The first path's bandwidth drops to 0.5Mbps then restore to 2Mbps. The change takes 2s as a cycle, and each bandwidth lasts for 1s. The second path's bandwidth has the same pattern which drops to 1Mbps then restore to 2.5Mbps; 2) The first path's bandwidth drops to 0.5Mbps then restore to 2.5Mbps and the second path's bandwidth drops to 1Mbps then restore to 3Mbps. To investigate how the RTT difference influence the performance under dynamic network setting, we run under two RTT setting: 1) The minimum RTT of the two paths are 10ms and 40ms; 2) The minimum RTT of the two paths are 10ms and 80ms.

As Figure 11 and Figure 12 show, when the bandwidth varies, DAMS-Robust performs the best when compared with the other three algorithms. DEMS performs the worst as the average bandwidth is limited compared with the video bitrate. DAMS performs better than DAMS-C in most cases and the opposite occurs under the dynamic network setting 2 with



(a) Block completion rate of all blocks. (b) High priority block completion rate.

Fig. 12. Overall performance under dynamic network setting 2.



(a) Under dynamic network setting 1. (b) Under dynamic network setting 2.

Fig. 13. Block completion rate efficiency under dynamic network.

larger bandwidth variation and the RTT of two paths are 10ms and 40ms (left part of Figure 12(a)). Previous evaluation results in V-C1 show that DAMS-C performs better under small RTT difference than large RTT difference. Besides, as mentioned in Section III-C, DAMS-C has the highest slack, so it shows high tolerance for network variation. As a result, DAMS-C performs better than DAMS in this situation.

We show the efficiency of block completion rate in Figure 13. We design DAMS-Robust to reduce the bandwidth resource waste by reducing the overdue data arriving at the receiver caused by the unknown network variation. The efficiency here refers to the ratio of blocks arrive before the deadline to all received blocks. As the Figure 13(a) and Figure 13(b) show, DAMS-Robust exhibits greater or nearly equal efficiency than DAMS. We can also find that the efficiency of DAMS-C is high, however, the total completed blocks of DAMS-C is lower than DAMS-Robust.

VI. Discussion

Dependency effect. In this paper, we focus on the number of blocks that meet the deadline. In Section V, we showed that DAMS achieves the highest block completion rate under various network conditions. A natural question is to extend our design to multiple blocks with dependency among them as dependency may exist in some applications. We can increase the capability of DAMS and perform further processing to deal with dependency of blocks. One direction is to simply cancel the blocks which depend on the canceled blocks. However, the cancelled blocks may increase a lot. More complex algorithm to deal with dependency tree of blocks could be an option and we leave this as future work.

Path extension. Our evaluations are all conducted with consideration of two paths as it is a most common case today. However, our design of DAMS, which achieves finishing sending simultaneously can also be extend to more than two paths without much additional design. If we need to move the exceed data from the path which completes late to other paths, we can only choose the other path in two paths' condition. If more paths exist, we can choose in order of RTT and prioritize the way with the smallest RTT to put data that exceeds the deadline. If this way cannot make these data meet the deadline, we can choose the way with the second smallest RTT.

Loss awareness. Packet loss should be taken into account when determining whether or not the block will meet the deadline. If the packet loss rate is non-negligible and affects the block completion time, forward error correction (FEC) is a choice to improve the block completion rate, whose ratio can be configured according to the packet loss rate. Under network conditions with packet loss, the block completion time can be estimated with the analysis of retransmission mechanism. The estimated block completion time facilitates the scheduling of blocks on the server to improve the block completion rate. However, dealing with packet loss is not the focus of this paper and we leave it as the future work.

VII. Conclusion

In this paper, we proposed DAMS, a deadline-aware multipath scheduler for streaming blocks aiming to meet the blocks deadlines. DAMS is aware of the deadline and judiciously handle the sending order of blocks with heterogeneous attributes. Besides, DAMS finishes the sending on multiple paths simultaneously for each block at the sender to reduce the sunk cost of data block transmission and avoid bandwidth waste brought by preemption of unknown blocks. Through extensive evaluations, we found that DAMS outperforms existing multipath schedulers by 41%-63% on average.

VIII. Acknowledgement

This work was supported by National Key R&D Program of China (No. 2018YFB1800303) and NSFC (No. 6213000078 and No. 61872211). Dr. Xin Wang's work was supported by NSF ECCS 2030063, OIA 2134840.

References

[1] M. Baldi and Y. Ofek, "End-to-end delay analysis of videoconferencing over packet-switched networks," *IEEE/ACM Transactions On Networking*, vol. 8, no. 4, pp. 479–492, 2000.

[2] H. Shi, Y. Cui, F. Qian, and Y. Hu, "Dtp: Deadline-aware transport protocol," in *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*, 2019, pp. 1–7.

[3] Y. Cui, Z. Liu, H. Shi, J. Zhang, and K. Zheng, "Deadline-aware transport protocol. internet draft draft-shi-quic-dtp-02," 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-shi-quic-dtp/

[4] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "Mp-dash: Adaptive video streaming over preference-aware multipath," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 129–143.

[5] H. Shi, Y. Cui, X. Wang, Y. Hu, M. Dai, F. Wang, and K. Zheng, "Stms: Improving mptcp throughput under heterogeneous networks," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 719–730.

[6] "The first multipath tcp enabled smartphones." 2018. [Online]. Available: http://blog.multipath-tcp.org/blog/html/2018/12/10/the_first_multipath_tcp_enabled_smartphones.html

[7] "Use multipath tcp to create backup connections for ios." 2017. [Online]. Available: https://support.apple.com/en-us/HT201373

[8] E. Dong, M. Xu, X. Fu, and Y. Cao, "A loss aware mptcp scheduler for highly lossy networks," *Computer Networks*, vol. 157, pp. 146–158, 2019.

[9] H. Lee, J. Flinn, and B. Tonshal, "Raven: Improving interactive latency for the connected car," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 557–572.

[10] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer, "Musher: An agile multipath-tcp scheduler for dual-band 802.11 ad/ac wireless lans," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[11] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "Fmtcp: A fountain code-based multipath transmission control protocol," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 465–478, 2014.

[12] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath {TCP}," in *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 399–412.

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

[14] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet, "Design, implementation, and evaluation of energy-aware multi-path tcp," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–13.

[15] Y. Go, O. C. Kwon, and H. Song, "An energy-efficient http adaptive video streaming with networking cost constraint over heterogeneous wireless networks," *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1646–1657, 2015.

[16] A. Ford, C. Raiciu, M. Handley, O. Bonaventure *et al.*, "Tcp extensions for multipath operation with multiple addresses," RFC 6824, Tech. Rep., 2013.

[17] Y.-s. Lim, E. M. Nahum, D. Towsley, and 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.

[18] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 141–153.

[19] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream control transmission protocol," 2007.

[20] J. Wu, B. Cheng, and M. Wang, "Improving multipath video transmission with raptor codes in heterogeneous wireless networks," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 457–472, 2017.

[21] B. Jonglez, M. Heusse, and B. Gaujal, "Srpt-ecf: challenging round-robin for stream-aware multipath scheduling," in *Second Workshop on the Future of Internet Transport (FIT 2020)*, 2020.

[22] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[23] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 99–114.

[24] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[25] T. P. Baker, "An analysis of edf schedulability on a multiprocessor," *IEEE transactions on parallel and distributed systems*, vol. 16, no. 8, pp. 760–768, 2005.

[26] M. Bertogna and S. Baruah, "Tests for global edf schedulability analysis," *Journal of systems architecture*, vol. 57, no. 5, pp. 487–497, 2011.

[27] Y. Sun, G. Lipari, N. Guan, and W. Yi, "Improving the response time analysis of global fixed-priority multiprocessor scheduling," in *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 2014, pp. 1–9.

[28] G. Tong and C. Liu, "Supporting soft real-time sporadic task systems on uniform heterogeneous multiprocessors with no utilization loss," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2740–2752, 2015.

[29] K. Yang and J. H. Anderson, "On the soft real-time optimality of global edf on multiprocessors: From identical to uniform heterogeneous," in *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 2015, pp. 1–10.

[30] J.-J. Han, S. Gong, Z. Wang, W. Cai, D. Zhu, and L. T. Yang, "Blocking-aware partitioned real-time scheduling for uniform heterogeneous multicore platforms," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 1, pp. 1–25, 2020.

[31] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 1–15.

[32] R. Fontugne, C. Pelsser, E. Aben, and R. Bush, "Pinpointing delay and forwarding anomalies using large-scale traceroute measurements," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 15–28.

[33] J. Iyengar and M. Thomson, "Quic: A udp-based multiplexed and secure transport draft-ietf-quic-transport-23," 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-quic-transport-23

[34] "tc: Linux advanced routing and traffic control." http://lartc.org/lartc.html.

[35] C. G. Bampis, Z. Li, I. Katsavounidis, T.-Y. Huang, C. Ekanadham, and A. C. Bovik, "Towards perceptually optimized end-to-end adaptive video streaming," *arXiv preprint arXiv:1808.03898*, 2018.

[36] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.