DMTP: Deadline-aware Multipath Transport Protocol

Tony John
Networks and Distributed Systems Lab
OVGU Magdeburg

Adrian Perrig
Network Security Group
ETH Zürich

David Hausheer
Networks and Distributed Systems Lab
OVGU Magdeburg

***Abstract*—Increasing demand for real-time applications such as teleoperation, remote control of a process, and live video streaming has led to the need for more efficient and reliable transport protocols that can satisfy strict latency requirements. Emerging multipath communication can help to achieve such goals. However, existing multipath protocols, such as Multipath TCP and Multipath QUIC fall short on real-time applications, as they rely on retransmissions and congestion control, leading to higher latencies on lossy links. This paper proposes DMTP, a new deadline-aware multipath transport protocol based on the SCION Internet architecture to support real-time applications with strict latency requirements. Our evaluations show that DMTP outperforms Multipath TCP and Multipath QUIC over lossy links by more than 40% in terms of the number of packets transferred within their deadline when the loss rate is above 2%.**

***Index Terms*—Multipath communication, deadline-aware transport protocol, SCION Internet Architecture**

I. Introduction

With the advent of the Internet of Things and Industry 4.0, the number of devices connected to the Internet continues its steep ascent. Along with their number, the capabilities and functionalities of devices are also increasing [1]. A subset of these devices provides real-time applications such as teleoperation [2], remote process control, or live video streaming [3]. Such applications share a common characteristic: the traffic they send is latency-sensitive, and often has strict deadline requirements, which means that the data must be received and processed within a specific time frame; otherwise, it becomes irrelevant. Other requirements can vary among these applications. Live video streaming, for example, can suffer a few dropped frames as long as the liveness of the video is maintained. On the other hand, remote process control typically requires both reliability and recency of the data. Sometimes, multiple real-time applications run in parallel and they need to handle multiple data streams with different requirements. Another application area that requires strict deadline adherence is competitive online gaming. Periods of higher latency than other players can result in a disadvantage to a player. An example of this happened in the 24 hours of Le Mans Virtual, where a professional racecar driver (Max Verstappen) lost the race due to network lag [4]. In the realm of real-time multiplayer games, the impact of delay on user experience and gameplay is a critical consideration [5].

ISBN 978-3-903176-57-7

At the same time, the connectivity available to Internet devices today is evolving with technologies such as 5G and low-earth-orbit satellite constellations such as Starlink that provide low-latency communication worldwide. As many Internet-connected devices are multihomed and have multiple interfaces, *e.g.*, smartphones with 5G and WiFi-based connections, a protocol that utilizes these multiple heterogeneous paths can be beneficial to latency-sensitive applications. Consider a sample device with access to two paths: one path features high bandwidth, but is lossy and has high latency, while another path has low bandwidth, low latency, and low loss. In this case, the high-bandwidth path could be used for most of the traffic, and the low-latency path for retransmissions and acknowledgements. Here the high-bandwidth link is utilized despite being lossy, providing higher throughput for the application while still satisfying the latency requirement. However, multihoming provides only limited optimization potential, since the paths in today's Internet cannot be controlled by the end-user. New Internet architectures such as SCION [6] provide end-hosts with the capability of choosing the path(s) to use for communication and supports communication over multiple paths. With SCION's path-aware forwarding, applications can choose a set of paths that suit their needs based on a set of available paths.

The topic of multipath transport has recently gained renewed attention. Research in this diretion has already been conducted in the field of multipath protocols with Multipath TCP (MPTCP) [7]–[10] and Multipath QUIC (MPQUIC) [11]–[14]. The aim of these protocols is to transport data reliably such that no packets are discarded. The receiver acknowledges the received packets, and the sender retransmits packets if necessary, leading to higher latency on lossy links. This is suitable for applications where reliability is critical, but unsuitable for real-time applications as these protocols lack strict latency guarantees. There have been attempts to combine Forward Error Correction (FEC) with MPTCP and MPQUIC to support latency-sensitive applications [10], [15]. However, these attempts do not support path-aware networks like SCION but instead rely on multihomed devices which are limited in their path selection.

Therefore, this paper proposes DMTP, a deadline-aware multipath transport protocol based on the SCION Internet architecture. DMTP enables application developers to create an end-to-end tunnel with hard latency requirements between two

nodes for real-time communication. Our protocol minimizes late packet arrival in two ways: intelligent packet retransmission and FEC. Intelligent packet retransmission ensures that retransmission occurs only when that packet can reach the receiver within the specified deadline. FEC adds redundancy packets so that some packets can be reconstructed at the receiver in case of packet loss. Additionally, the protocol has an optimization engine that can minimize the cost of transmission through dynamic optimal path selection and optimal distribution of data over those paths. Utilizing the path-aware routing of SCION, our protocol monitors the available paths and selects the optimal combination of paths with which the deadline and reliability requirements of the application can be satisfied with high probability while minimizing costs. However, the scope of this paper does not include an evaluation of optimal path selection. Instead, it focuses on the effective utilization of a set of preselected paths.

II. Problem description

The data flows in applications with strict latency requirements, such as teleoperation of machinery, videoconferencing or remote control of industrial control systems, has a common characteristic. The traffic must reach the receiver within a specified timeframe. After that timeframe passes, the data is no longer relevant. Therefore, this paper aims to develop a deadline-aware multipath transport protocol, utilizing multiple paths and FEC to satisfy deadline requirements with a high probability while minimizing cost. In the context of this work, the term cost specifically refers to the expenses associated with transmitting data through a particular path. To this end, given a set of paths, packet delivery importance and a reception deadline, we define our problem as follows:

- Find an optimal combination of paths with the minimum overall cost that can satisfy the reception deadline and the overall bandwidth required to send the data streams.
- Find an adaptive FEC coding rate that can combat the losses on the selected paths by using the minimum number of redundancy packets while continuously adapting to changes in the loss rate.
- Determine a sending schedule that distributes redundancy and data packets among the selected paths and follows the packet delivery importance.
- Find a suitable retransmission threshold after which lost packets are retransmitted if the reception deadline can be satisfied.

III. Related work

Works related to deadline-aware transport protocols can be broadly divided into three areas: single-path approaches, multipath approaches, and FEC-based approaches.

A. Single-path Approaches

Shi et al. [16] introduce the Deadline-Aware Transport Protocol (DTP), which is an extension to the QUIC [17] transport protocol. DTP's deadline-aware scheduler tries to deliver blocks within the specified deadline. DTP takes into account the time-sensitive nature of real-time traffic and prioritizes packets based on their deadline. The protocol also uses a dynamic window mechanism that adjusts the rate at which packets are sent based on network conditions. Additionally, DTP supports multiplexing of different streams with different priorities and deadlines [16].

Unlike DTP, our protocol has a different design approach where we utilize multiple paths and FEC to satisfy deadline requirements with a high probability while minimizing cost. Therefore, we are not comparing our protocol with DTP.

B. Multipath Approaches

Chuat et al. [18] investigate how to optimize communication in a path-aware multi-path network to meet a deadline-constrained communication demand. The paper presents an optimization problem for multipath communication, which takes into account the path metrics, the communication demand, and the deadline constraints. The notion of FEC is not introduced in this work. The paper also provides a heuristic algorithm to efficiently solve the optimization problem. This algorithm is adapted in our protocol for selecting the optimal path combinations.

MPTCP [19] extends TCP to support the utilization of multiple paths between endpoints. This is achieved by splitting a single connection into multiple subflows, each utilizing a separate path, and reassembling the data at the receiver. MPTCP uses mechanisms such as subflow management, congestion control, and data mapping to coordinate the multiple paths and provide a transparent and seamless experience to the application layer. MPTCP has been widely researched and evaluated in various scenarios and has been shown to provide significant improvements in terms of reliability, performance, and energy efficiency. MPTCP suffers from some of the drawbacks of TCP. It can have head-of-line blocking at the subflow level and the acknowledgements need to be sent through the same path that data was sent. These drawbacks make MPTCP unsuitable for deadline-aware communication use cases.

Nguyen et al. [20] propose a method for improving the performance of MPTCP. The authors argue that MPTCP's performance can be limited by its lack of path awareness, i.e., it does not take into account the differences between the various paths available for transmission. To address this issue, the authors propose a path-aware approach to reinforce MPTCP by incorporating information about the characteristics of each path into the MPTCP protocol. Similarily, Dong et al. [9] propose a loss-aware MPTCP scheduler, which prioritizes subflows based on their current loss rate, in order to enhance the performance of data transmission in highly lossy networks. The scheduler monitors the loss rate of each subflow, and dynamically adjusts the priorities of subflows based on the loss rate. The path-awareness presented in these two works provides more information on paths and improves the scheduling in MPTCP. But it is different from path-aware properties of Internet architectures like SCION. They are still

limited to multihomed devices and cannot utilize multiple inter-domain paths as available through SCION.

De Coninck et al. [21] present MPQUIC, an extension of the QUIC protocol which enables the use of multiple paths for transmitting data between client and server to increase performance, reliability, and resilience. The authors were able to show that MPQUIC outperforms QUIC, TCP and MPTCP in high bandwidth-delay product and lossy links for longer data transfers. However MPQUIC suffers timeouts and retransmission delays at higher loss rates, making it unsuitable for latency-sensitive applications.

XLINK [12] provides a framework for MPQUIC transport in large-scale video services, which can be used to optimize the performance of the network with respect to Quality of Experience (QoE). The proposed framework consists of a scheduling algorithm, an accurate video streaming QoE model and a smart packet reinjection algorithm to prevent head of line blocking and reduce loading times. XLINK primarily focuses on on-demand video streaming to multi-homed mobile devices with a WiFi and a cellular connection. An adaptation of the smart packet reinjection presented in this work is utilized in our protocol.

As existing multipath approaches tailored to latency-sensitive applications are mainly adaptations of MPTCP and MPQUIC, we are evaluating our protocol against them. Unlike these adaptations, our protocol supports path-aware networks utilizing SCION and uses a combination of path selection, adaptive FEC and smart packet retransmission.

C. FEC-based Approaches

FEC-based approaches can significantly improve deadline-aware communication by introducing redundancy in the data transmission, allowing for the recovery of lost or corrupted packet without retransmission. Its addition can complement the approaches mentioned previously as it provides an additional layer of reliability and reduces delays due to retransmissions.

Ferlin et al. [10] present an approach to support latency-sensitive applications over heterogeneous networks by combining MPTCP and FEC. The approach is based on a client-side adaptation of MPTCP that dynamically switches between different paths combined with an FEC-based congestion control algorithm that adaptively adjusts the sending rate based on feedback from the FEC receiver. However it does not offer deadline guarantees and is limited to multihomed devices.

Vu and Wolff [15] discuss how FEC can be used as a mechanism to support delay-sensitive applications, such as video streaming, over MPQUIC [11]. The scheme uses a combination of redundancy and interleaving techniques to minimize the impact of packet losses. The results show that the proposed scheme can improve the quality of video streaming over an MPQUIC network. While this approach minimizes the impact of packet losses, it does not offer deadline guarantees.

Additionally, there exist also adaptive FEC-based approaches where the coding rate dynamically adjusts to the network conditions to improve transmission efficiency.

LightFEC [22] presents a new approach to FEC in networks using a deep-learning technique. The proposed method is designed to be lightweight and adaptive to changing network conditions. The paper argues that traditional FEC methods are ill-suited for dynamic network conditions, and proposes a deep-learning based solution to overcome these limitations.

DeepRS [23] presents another approach to FEC in real-time video communications using deep learning. The authors propose a network-adaptive FEC that uses a deep neural network to predict the rate of errors in the network and adjusts the FEC accordingly. The system aims to improve the efficiency of error correction for real-time video communications in networks with changing loss rates.

These two works show that an adaptive FEC algorithm leads to a more efficient use of the available network resources than a fixed FEC approach. This is essential for minimizing overhead and transmission cost, which are objectives of our protocol. Our protocol employs a simpler algorithm for adaptive FEC and we are aiming to improve it using neural network based approaches like the two aforementioned works in the future.

IV. SCION Internet Architecture

SCION [6] is a secure next-generation Internet architecture that provides route control, fault isolation, and explicit trust information for end-to-end communication. One of the key features of SCION is path-aware networking, where the end hosts can choose the end-to-end paths, including all the inter-domain hops. It enables several optimization possibilities for the end user, such as choosing the optimal path or a combination of paths for a specific application based on metrics such as latency, jitter, bandwidth and cost. SCION opens up possibilities of multipath communication beyond multi-homed devices. It enables the use of multipath protocols even on singly connected endpoints.

SCION introduces the concept of Isolation Domains (ISD), which is a collection of Autonomous Systems (AS). An ISD can represent domains such as, a geographical area, legal entity or an organization. To govern an ISD, one or more ASes in it forms the ISD core which provides connectivity to other ISDs and manages the cryptographic trust root (TRC) of that ISD. Routing between ASes within an ISD is isolated within that ISD. Unlike today's Internet, routing in SCION follows the packet carried forwarding state (PCFS) methodology. Every SCION packet will have the complete inter-AS path within its header in the form of hop fields. Through the dissemination of path-segment construction beacons (PCBs) from the core ASes, paths segments are explored and end-to-end paths are created by combining these path segments.

Figure 1 illustrates a SCION network with four ISDs and a collection of ASes within each ISD. The blue and red paths shows two of the possible end-to-end paths between AS C and AS G.

There are several real-world deployments of SCION worldwide, provided by eight Internet Service Providers [24]. These deployments are running in parallel to today's Internet. For

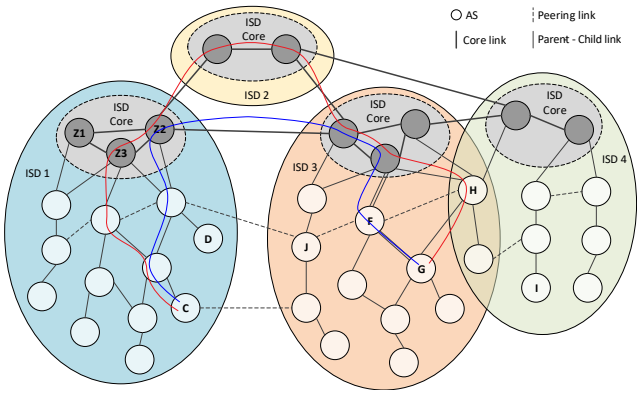

Fig. 1. SCION network with four ISDs showing two of the possible end-to-end paths between ASes C and G

research and running experiments on a SCION network, SCIONLab provides a global SCION testbed where users can create personal ASes [25].

V. Deadline-Aware Multipath Transport Protocol

The goal of our Deadline-Aware Multipath Protocol is to deliver data from one host to another within a specified deadline with minimum overhead and minimum cost. Throughout the rest of the paper, the term protocol refers to the Deadline-Aware Multipath Transport Protocol. Our protocol runs in the user space and uses the path-aware networking of SCION Internet Architecture for communication. As UDP is the defacto transmission protocol upon which most real-time and latency-sensitive protocols are built, we are using the SCION version of UDP, SCION-UDP as our base transmission protocol.

To fulfill the deadline and reliability requirements, we incorporate the following technologies into a transport protocol:

- Optimal path combination selection
- Smart packet retransmission
- Adaptive FEC

Chuat et al. [18] describe the optimal path selection for deadline-sensitive data as an optimization problem. They have proposed to use linear programming (LP) to select the optimal paths and the proportion of bandwidth to assign to each path while maximizing communication quality or minimizing cost. We are adopting this strategy in our protocol. The heuristics-based approach requires less computation compared to solving the LP problem, making it more suitable for scenarios with limited computational resources or in highly dynamic networks where path selection needs to be run at a higher frequency. However, the heuristics-based approach might not always find the optimal solution, as it relies on approximations, while the LP-based approach guarantees optimality under certain conditions. Thus, depending on the specific network requirements and constraints, either the heuristic algorithm alone may be sufficient, or a combination of LP and heuristics may be needed for optimal path selection. Alternatively, the Path-Aware Networking Application Programming Interface (PANAPI) [26] may also be used for optimal path selection.

Smart packet retransmission ensures that retransmission happens only if the retransmitted packet can reach the receiver within the required deadline and retransmission happens through one of the fastest paths available.

FEC works by generating and sending redundancy packets along with data packets to recreate packets lost in transmission. It utilizes additional bandwidth to avoid retransmission. We have opted to use Reed-Solomon coding [27] as our FEC method as it supports multiple loss recovery. Our protocol adjusts the coding rate according to the observed loss on each path used.

A. Protocol Architecture

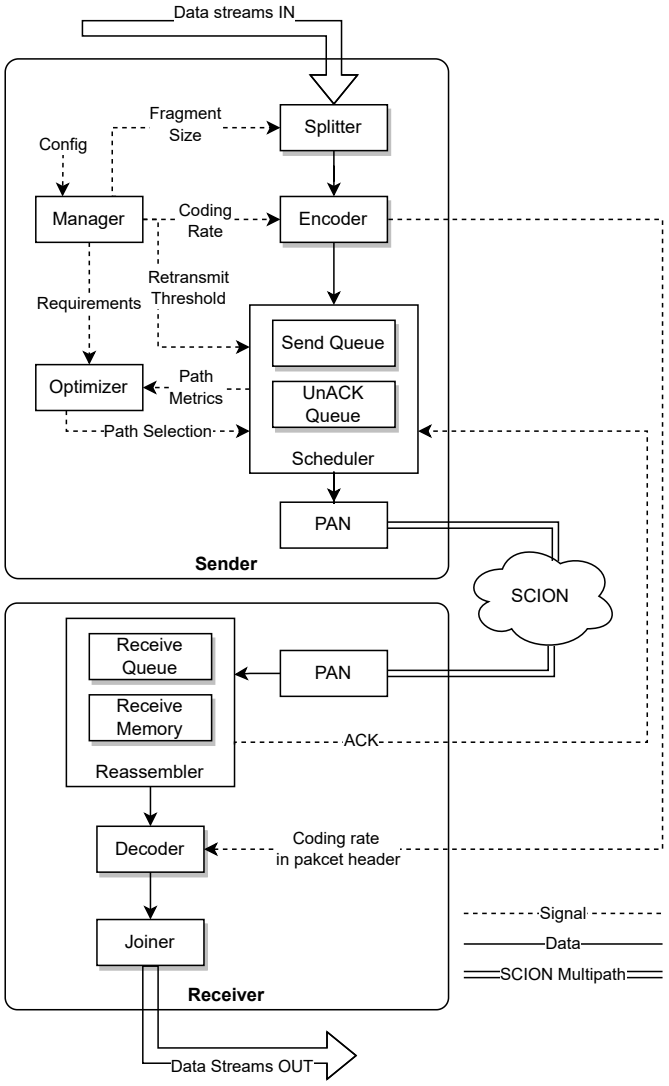

Fig. 2. Protocol Architecture

Figure 2 shows the high-level architecture of our protocol. The two nodes communicating with each other will have a sender and receiver component. In the figure, only one sender and receiver are shown for simplicity. The sender takes in multiple streams of data and sends it to the receiver over

multiple available paths such that it reaches the receiver within the specified deadline. For a particular incoming stream of data, the first module that it encounters is the splitter. A block of data written to the sender is termed a frame, and each frame is given a monotonically increasing frame ID. A frame is then split into equally sized chunks called fragments. The size of the fragments can be set to maximize the utilization of the available MTU of the selected paths. Fragments then go through the encoder. Using Reed-Solomon coding, it adds redundancy fragments to that particular frame. The encoded frame is then added to the send queue. The scheduler takes the frame from the send queue, adds a header to its fragments and assigns them to different paths specified by the optimizer. The scheduler also transfers that frame to the unacknowledged (unack) queue. Each incoming stream has its own send queue and unack queue.

At the receiver side, the received fragments are added to the receive queue. The receive queue can handle fragments coming in out of order. All the information needed for reassembly is provided in the header, which will be discussed later in this section. When a frame in the receive queue has enough fragments to be decoded, it is then decoded, and fragments are joined to form the original frame. The joined frame is then sent out. The receiver also takes the header from the received packets and sends them back through the acknowledgement path as acknowledgements. The receive memory enables dropping duplicate fragments.

In the following paragraphs, the core mechanisms of our protocol are described in detail.

Packet header: Figure 3 depicts the 8-byte header added to packets by the protocol, containing all the information for the receiver to identify the stream it belongs to, the frame ID, the number of fragments in the frame, the number of redundancy fragments and the amount of padding added to the fragment.

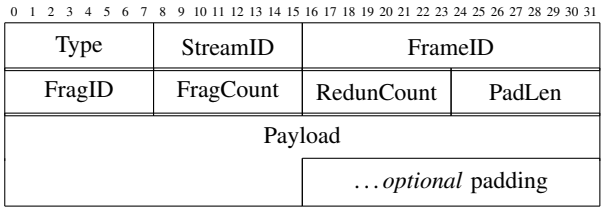

Fig. 3. Header description

Optimizer: The optimizer takes in costs, available bandwidth, one-way delay and loss of the paths available and the deadline and bandwidth requirements of the data that needs to be sent. As output, the optimizer provides the paths to use, the fraction of bandwidth to assign to each path and the retransmission path. The optimizer ensures that the selected paths, in combination with the retransmission path, can satisfy the deadline requirement while minimizing the transmission cost. The optimizer uses a combination of linear programming and heuristics [18] to determine the best path selection and bandwidth allocation.

Retransmission: Let δ be the specified deadline, d_{min} be the one-way delay of the retransmission path and d_{max} be the one-way delay of highest latency path among the selected paths. The maximum time it takes for an acknowledgment to reach the receiver will be $d_{max}+d_{min}$. If there is loss, then the time it will take for the retransmitted packet to reach the receiver will be $d_{max}+2d_{min}$. This value needs to be less than the specified deadline δ. So for a retransmission to be effective, it should happen right after $d_{max}+d_{min}$. This ensures that there is enough time for all acknowledgements to be received and time left to retransmit lost packets within the deadline. That is the retransmission threshold. To this effect, a timer is attached to each frame in the unack queue with its timeout being the retransmit threshold. After the timeout, the scheduler checks if the acknowledged fragments are enough to decode the whole frame. If not, the unacknowledged fragments are retransmitted through the retransmit path. This mechanism ensures that there are no unnecessary retransmissions, and retransmission happens only when the deadline can be satisfied. Figure 4 shows a sequence diagram of the retransmission mechanism while delivering a frame having five fragments. In this case, the optimizer selected Path 1 for transmitting all fragments and selected the faster Path 2 for acknowledgments and retransmissions. Fragments 2 and 4 were lost in flight. Sender knows about the lost fragments from the acknowledgements received through Path 2. It waits till the restransmit threshold and sends the lost packets again through the faster Path 2.

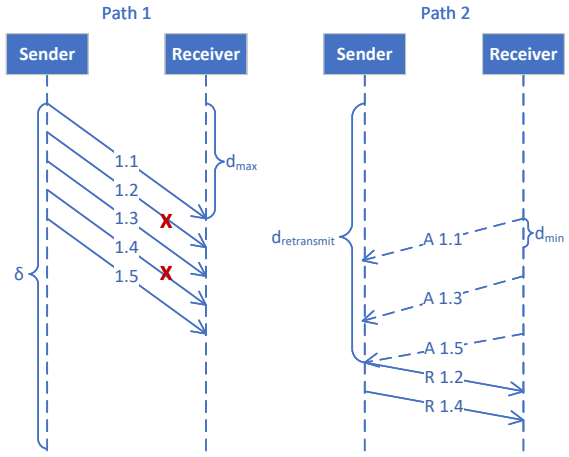

Fig. 4. Delivery of a frame with five fragments

Adaptive FEC: FEC trades additional bandwidth to avoid retransmissions. Minimizing retransmissions can improve the transmission delay beyond the specified deadline. However, having a higher coding rate will lead to higher overhead in terms of required bandwidth and computational resources. Ideally, the coding rate should be just enough to overcome the loss. In a dynamic environment such as the Internet, the coding rate should be adapted according to the changes in loss rate on paths to achieve higher efficiency. Another factor to consider is the correlation of losses to additional bandwidth introduced by the FEC [18]. We can de-correlate losses by sending redundancy fragments over another path. Yu et al. [28]

demonstrate that in a network with multiple heterogeneous disjoint paths, sending redundancy packets and data packet through different paths reduces the effect of loss correlation on FEC performance. Our protocol's scheduler follows this and sends redundancy fragments over a different path than the one majority of the data fragments are sent on.

Reed-Solomon coding used in our protocol supports multiple erasures, enabling granular control over the coding rate. Losses over the selected paths are calculated for every frame from the acknowledgements, and the encoder stores a moving window of previous losses measured. The max loss in this window determines the coding rate for each frame. The upper limit of the coding rate can be configured in the protocol to limit the amount of additional bandwidth. In special cases where the stream's required bandwidth is low and the required reliability is high, a coding rate of 1 can be chosen, which duplicates the data over two paths. Our protocol allows users to set different coding rate limits to different streams.

Interaction between retransmission and adaptive FEC: Adaptive FEC requires the acknowledgements from a frame to adapt to a change in the loss rate. This change can only be made to the next frame. In this case and in cases where the loss was higher than the number of redundancy fragments, retransmission kicks in and ensures that the lost fragments are retransmitted. Together, these two mechanisms work in tandem to provide a robust and resilient data transmission. Overall, the interaction between retransmission and adaptive FEC is a key aspect of ensuring reliable and deadline bound data transmission. By adjusting the settings for retransmission and adaptive FEC, the system can be optimized to meet the specific requirements of the application. For example, the coding rate can be limited to a lower value to rely more on retransmission in case of loss.

Multiplexing multiple streams with various requirements: The protocol is designed in such a way that it supports multiple streams with varying quality and reliability requirements. In addition to the normal mode of operation, utilizing a combination of smart retransmission and adaptive FEC, streams can be of the following modes: FEC only, smart retransmission only, no FEC and no retransmission and finally, duplication of data over multiple paths for very high reliability. The last mode is only suitable for streams with low bandwidth requirements. Additionally, a priority can be added to each stream. The protocol should be configured with the number of streams required and their desired properties during startup. The encoder and the scheduler treat the frames according to their streams' configuration. The scheduler multiplexes the streams to the selected paths following the priority assigned to each stream.

Measuring path metrics: SCION's beaconing provides the provision for publishing the latency and bandwidth of path segments [6]. If this information is available, it is used as the starting point for running the optimizer. If not, the protocol assumes that the paths are symmetric and measures the round trip times of the available paths through SCMP [6] messages. During run-time, our protocol sends probe packets at regular intervals through the selected paths to calculate accurate latency of those paths. Initially, the available bandwidth of available paths is measured using probe packets following the probe gap model. In the current version of our protocol, we assume that all paths are disjoint. If high loss is observed on a path due to congestion, either the optimizer will choose another path or if no other path is available, the protocol will inform the application that the overall available bandwidth is lower and the application can choose to lower the sending rate. Once the initial path metrics are collected, the optimizer can select the optimal set of paths. Since acknowledgements are sent on a different path, it is difficult to accurately measure the one way delay of each path. So, probe packets are sent at regular intervals over the selected paths to measure their round trip time. Suppose there are more paths available after the path selection. In that case, at least two more paths chosen randomly are probed periodically so that the optimizer will have their metric in case of a path failure or significant increase in loss or latency on one or more of the chosen paths. The loss rate is measured from the acknowledgements of each frame sent. Since the scheduler knows the path each fragment was assigned to, the loss rate of all the selected paths can be calculated.

B. Prototype Implementation

A prototypical implementation of our protocol was developed in Go[1] programming language version 1.16. The Path-Aware Networking (PAN) library[2] was used in the implementation to communicate with the SCION end host stack. The prototypical implementation is developed in the form of a library and this library can be used to add our protocol to Go applications.

VI. Evaluation

A test setup was created to evaluate our protocol using two virtual machines (VM) running Ubuntu 22.04. Each VM hosts a SCION AS by running the open source SCION stack[3]. The virtual machines are hosted on a machine running AMD Ryzen 5 2600 processor, and 8GB of ram is assigned to each VM. Virtualbox[4] provided the virtualization. Two SCION paths with different properties were created between the two ASes. Both paths are limited to a bandwidth of 15Mbps and a latency of 60ms. Linux's Netem [29] is used to limit the bandwidth and induce losses on the paths. One of the paths is set as the low-cost path and the other is set as the high cost path. The gateways were built using our protocol and they are designed provide an IP tunnel between the two nodes. The data generator and sink are applications that only supports regular IP communication, showing that our protocol can support legacy IP application through gateways. The gateways were configured to have a deadline requirement of 200ms. The path parameters were chosen to show the behaviors of the protocol

[1]https://go.dev/
[2]https://github.com/netsec-ethz/scion-apps/tree/master/pkg/pan
[3]https://docs.scionlab.org/content/install/pkg.html
[4]https://www.virtualbox.org/

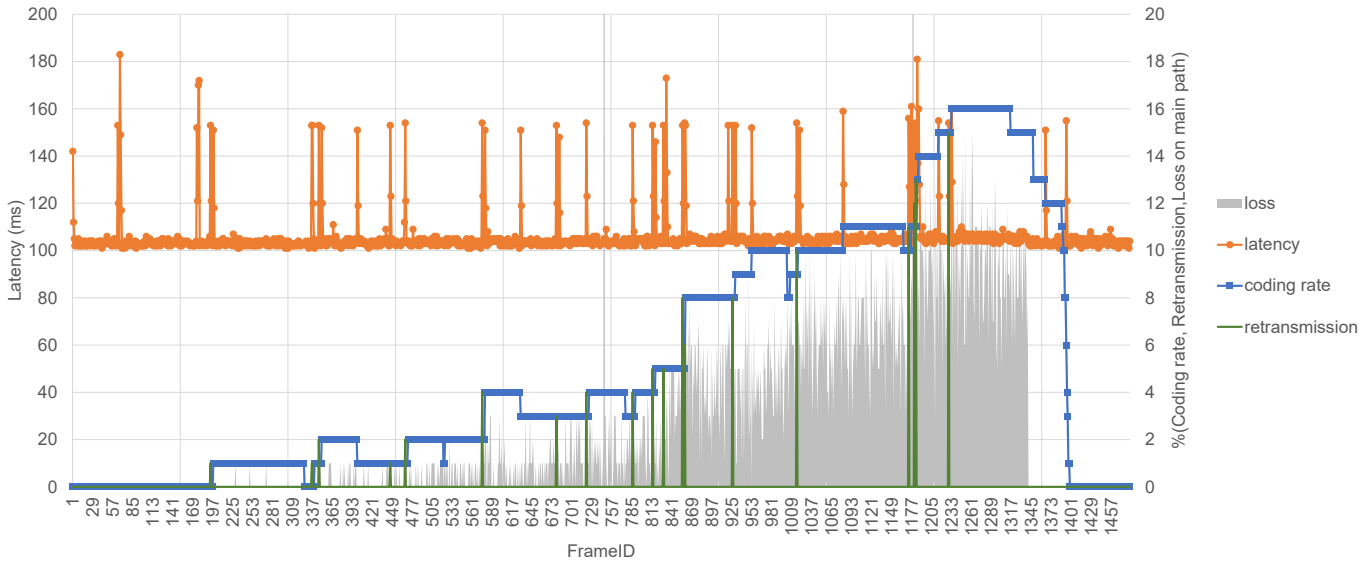

Fig. 5. Dynamic behaviour of DMTP

at varying loss rates on the low-cost path ranging from low levels to extreme levels. The low-cost and high-cost paths can also be considered as the main path and a backup path, where the backup path has a higher cost of transmission. Figure 6 illustrates the test setup.

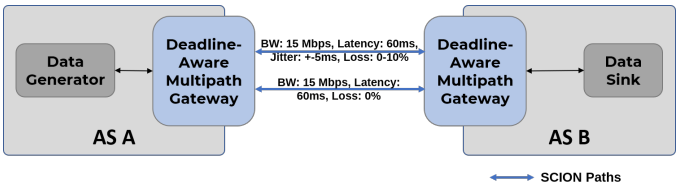

Fig. 6. Test setup

For each experiment, the data generator at AS A generates frames at a rate of 50 per second, with each frame having a size of 24KB. That makes the required bandwidth around 10Mbps. Each experiment lasted for 50 seconds. During the experiments, packet loss was induced on the low-cost path. increasing in rate every 5 seconds. The loss rate induced ranged from 0 to 10%. Figure 5 shows the dynamic behaviours of the gateway during the experiments. The protocol chose the low-cost path for most data transmission and the higher cost path for acknowledgements, retransmission and redundancy packets. Throughout the experiments, the gateways were able to maintain a deadline of less than 200ms. When it detected loss, retransmission was done and immediately, the coding rate was adapted to overcome the observed loss and avoid retransmission for the subsequent frames. We can see this behaviour whenever the loss rate changes significantly. Even at a loss rate of 10% on the high bandwidth path, the protocol was able to deliver the frames reliably within the specified deadline. With the combination of FEC and retransmission, all frames sent were received within the deadline without any frame loss.

Figure 7 shows the comparison of our protocol with MPTCP and MPQUIC. The MPTCP module bundled with the Linux kernel 5.15 and the default scheduler was used for this experiment. The number of frames reaching the receiver within the deadline decreases significantly above 0.5% loss rate when using MPTCP. For the MPQUIC experiment, the go implementation of MPQIUC[5] was used. This version of MPQUIC does not use FEC. MPQUIC performed better than MPTCP until 2% error rate. Above 2% loss rate, its performance was significantly degraded. Below 0.5% loss rate, MPTCP, MPQUIC and our protocol performed similarly.

Three versions of our protocol (only FEC, only smart retransmission and both combined) were tested under the different loss rates. As the protocol dynamically adjusts the coding ratio of its FEC, a few frames will be dropped when the loss rate changes suddenly in the FEC-only mode. But, the average latency of frames reaching the receiver was closer to the one-way delay of the main path being used which was the low-cost path. In the smart retransmission-only mode, at higher loss rates, the receiver waits for the lost packets to be retransmitted and this raised the average latency of the frames reaching the receiver. In this case no dropped frames were observed. And finally, using a combination of FEC and smart retransmission was tested. In this case spikes in latency were observed only when the FEC coding rate was being adapted and the average latency was closer to the one-way delay. The Combination of FEC and smart retransmission resulted in zero dropped frames even at a loss rate of 10%. Additionally, all frames reached the receiver within the specified deadline.

Figure 8 describes the bandwidth usage of the low-cost

[5]https://github.com/Abdoueck632/mp-quic/

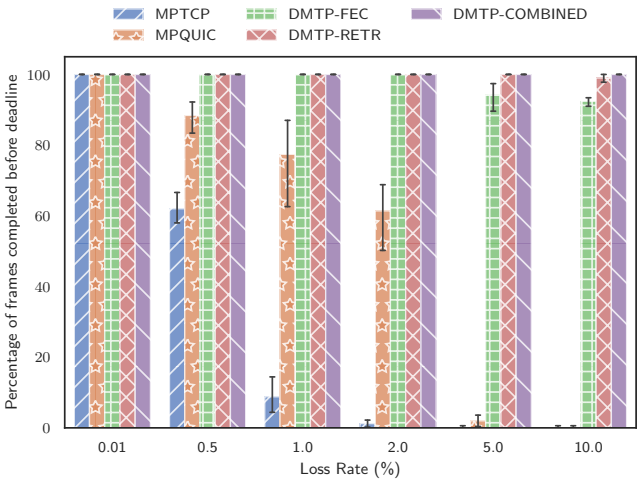

Fig. 7. Percentage of frames completed before the deadline under different loss rates.

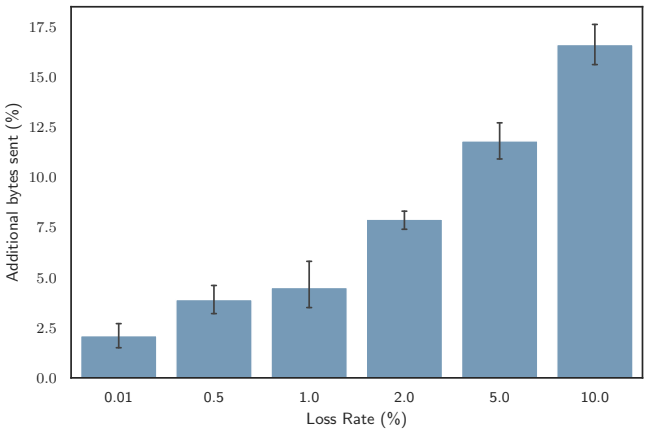

Fig. 9. Percentage of additional bytes sent at various loss rates

and high-cost paths for different loss rates on the low-cost path when a combination of FEC and smart retransmission is used. Even at higher loss rates, our protocol was able to utilize the low-cost path for sending the majority of the traffic while still satisfying the reception deadline and minimizing the overall transmission cost. Figure 9 describes the overhead of our protocol in terms of additional bytes sent. This includes the header added to each packet, retransmissions and redundancy packets sent. Despite the higher amount of additional bytes, DMTP always met the deadlines, even in scenarios with high loss rates, whereas the other candidates dropped a high number of frames.

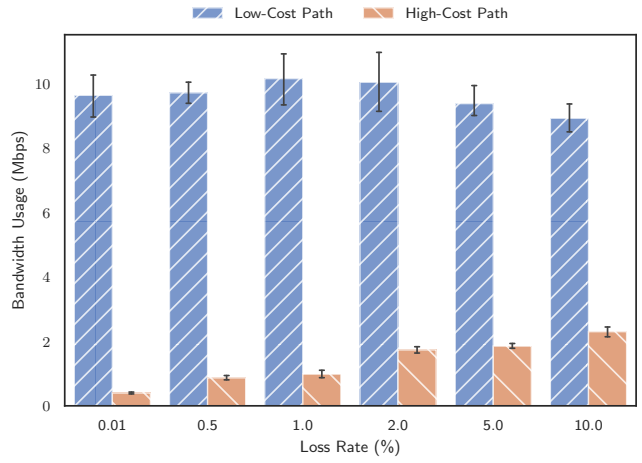

Fig. 8. Bandwidth usage of low-cost and high-cost paths at different loss rates.

VII. Discussion

The results of the evaluation demonstrate the effectiveness of DMTP. Under the test conditions, it was able to maintain the deadline of less than 200ms throughout the experiments, even when the loss rate was as high as 10%. The tests conducted with different configurations of our protocol (only FEC, only smart retransmission, and both combined) provided insights into the trade-off between FEC and smart retransmission. Based on these findings, the RETR option is best suited for situations with long deadlines and when low-latency paths are available, as it avoids the extra computational and bandwidth requirements of FEC. Conversely, an FEC-only approach is more effective when the required deadline is short and the path latencies do not support retransmissions within the deadline. For scenarios falling in between these two extremes, a combination of FEC and smart retransmission provides an optimal balance of performance and resource efficiency. In the cases with loss rates above 0.5% on the low-cost path, our protocol outperformed MPTCP and MPQUIC by a significant margin. Most of the frames sent through MPTCP and MPQUIC missed their deadline when losses were encountered and in those cases, DMTP was able to better utilize the available bandwidth and satisfy the deadline. These factors show the effectiveness of our protocol in latency sensitive use cases. Furthermore, our protocol offers the flexibility to adapt to varying application requirements and network conditions.

Besides the merits, DMTP has the following potential limitations:

- Assumptions: The protocol makes assumptions about the network conditions that may not hold in all cases, such as the availability of accurate metrics or the presence of disjoint paths.
- Deadlines may not be met in all cases: The protocol can ensure that the deadlines are met under the assumption that the path metrics measured are accurate and a combination of the available paths is able to satisfy the deadline requirement.
- Resource utilization: our protocol requires computation resources to run the optimization algorithm and to run the encoder and decoder of FEC, making it not suitable for significantly resource-constrained devices.

VIII. CONCLUSION AND FUTURE WORK

This paper aimed to develop a SCION-based transport protocol for latency-sensitive real-time applications capable of providing reliability and deadline guarantees utilizing multiple heterogeneous paths while minimizing cost. DMTP combines optimal path combination selection, smart packet retransmission and forward error correction to ensure that the deadline requirements of the application are satisfied with high probability while minimizing transmission cost. DMTP also supports multiple data streams with different latency and reliability requirements. The evaluation result shows that the proposed solution can effectively use multiple paths to maintain the required deadline without missing any frames at the receiver. The protocol performed well even at high loss rates and the results also showed that DMTP outperforms MPTCP and MPQUIC in a multipath network with lossy links.

Future work includes extending the path selection to include non-disjoint paths, passive measurement of available bandwidth and investigating the use of deep reinforcement learning for adaptive FEC and optimal path selection. Additionally, extensive evaluation of optimal path selection in a large topology, including the potential integration of PANAPI, will be conducted in future research to further improve our protocol's performance and capabilities.

ACKNOWLEDGMENT

We are grateful to Adrian Seiterle, Burak Cizmeci, Caspar Schutijser, Dominic Jud, Piet De Vaere, and Ralph Koning for their guidance and contributions in the development and refinement of DMTP.

REFERENCES

[1] N. Sharma, M. Shamkuwar, and I. Singh, "The history, present and future with iot," *Internet of things and big data analytics for smart generation*, pp. 27–51, 2019.

[2] J. S. Lee, Y. Ham, H. Park, and J. Kim, "Challenges, tasks, and opportunities in teleoperation of excavator toward human-in-the-loop construction automation," *Automation in Construction*, vol. 135, p. 104119, 2022.

[3] 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.

[4] G. Leporati, "An f1 champ lost a virtual race to lag. is the sport ready for primetime?" Jan 2023. [Online]. Available: https://www.washingtonpost.com/video-games/2023/01/23/sim-racing-verstappen-le-mans/

[5] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, 2002, pp. 23–29.

[6] L. Chuat, M. Legner, D. A. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, "The complete guide to scion-from design principles to formal verification," 2022.

[7] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath tcp: Analysis, design, and implementation," *IEEE/ACM Transactions on networking*, vol. 24, no. 1, pp. 596–609, 2014.

[8] 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.

[9] 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.

[10] S. Ferlin, S. Kucera, H. Claussen, and Ö. Alay, "Mptcp meets fec: Supporting latency-sensitive applications over heterogeneous networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2005–2018, 2018.

[11] 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.

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

[13] X. Shi, L. Wang, F. Zhang, B. Zhou, and Z. Liu, "Pstream: Priority-based stream scheduling for heterogeneous paths in multipath-quic," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–8.

[14] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2295–2310, 2020.

[15] V. A. Vu and J. Wolff, "Supporting delay-sensitive applications with multipath quic and forward erasure correction," in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2021, pp. 95–103.

[16] 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.

[17] 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.

[18] L. Chuat, A. Perrig, and Y.-C. Hu, "Deadline-aware multipath communication: An optimization problem," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2017, pp. 487–498.

[19] C. Paasch and O. Bonaventure, "Multipath tcp," *Communications of the ACM*, vol. 57, no. 4, pp. 51–57, 2014.

[20] K. Nguyen, M. Golam Kibria, K. Ishizu, F. Kojima, and H. Sekiya, "An approach to reinforce multipath tcp with path-aware information," *Sensors*, vol. 19, no. 3, p. 476, 2019.

[21] 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.

[22] H. Hu, S. Cheng, X. Zhang, and Z. Guo, "Lightfec: Network adaptive fec with a lightweight deep-learning approach," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3592–3600.

[23] S. Cheng, H. Hu, X. Zhang, and Z. Guo, "Deeprs: Deep-learning based network-adaptive fec for real-time video communications," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[24] C. Krähenbühl, S. Tabaeiaghdaei, C. Gloor, J. Kwon, A. Perrig, D. Hausheer, and D. Roos, "Deployment and scalability of an inter-domain multi-path routing infrastructure," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 126–140.

[25] J. Kwon, J. A. García-Pardo, M. Legner, F. Wirz, M. Frei, D. Hausheer, and A. Perrig, "Scionlab: A next-generation internet testbed," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–12.

[26] T. Krüger and D. Hausheer, "Towards an api for the path-aware internet," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration*, 2021, pp. 68–72.

[27] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[28] X. Yu, J. W. Modestino, and I. V. Bajic, "Modeling and analysis of multipath video transport over lossy networks using packet-level fec." in *DMS*. Citeseer, 2005, pp. 265–270.

[29] *tc-netem(8) — Linux manual page*, 2011.