# Improving QoE with Genetic Algorithm-based Path Selection for MPTCP

Shadi Bikas, Müge Sayıt, *Member, IEEE*

***Abstract*—Multipath TCP (MPTCP) is a protocol that enables the use of more than one subflow under the same TCP connection, which provides increased throughput due to the aggregated bandwidth. Therefore, the use of MPTCP can be very beneficial to video streaming applications since bandwidth is the most crucial parameter that improves the performance of such applications. However, the characteristics of the subflow paths might have a significant impact on application performance. In this study, we propose a path selection approach for MPTCP subflows in order to maximize the Quality of Experience (QoE) for adaptive HTTP streaming systems, which is currently one of the most popular video streaming application techniques. The paths are selected on the network layer by taking the bandwidth and delay differences into account. The disjointness of the paths is also considered in the proposed path selection approach. A genetic algorithm-based approach is utilized for the selection of the paths. Furthermore, we indicate how the characteristics of the paths affect QoE. The experimental results indicate that considering bandwidth and delay difference parameters jointly helps increase all QoE metrics, compared to the approaches that select paths considering either bandwidth or delay difference.**

***Index Terms*—Adaptive streaming, SDN, Routing, Genetic algorithm.**

## I. Introduction

VIDEO streaming applications are one of the most popular applications running over the Internet. The latest Global Internet Phenomena Report [1] states that the traffic produced by video streaming services dominates the Internet, accounting for 65.93% of total traffic in 2022. With the recent developments in video codec standards and the increasing demand of Internet users to play a video with low latency and high quality and resolution, the bandwidth requirements of video streaming applications tend to increase. One of the most well-known video streaming techniques is HTTP adaptive streaming (HAS). In such applications, there is more than one encoded video file with various qualities, where each file is partitioned into small and fixed-sized segments. The clients send a request for the segments in different qualities over time to adapt the video quality based on the observed network conditions and internal parameters. Clearly, bandwidth is the most important criterion that affects the Quality of Experience (QoE) in HAS applications.

S Bikas is with the International Computer Institute, Ege University, Izmir, Turkey. Email: shadi.bikas90@gmail.com.
M Sayıt is with Computer Science and Electronic Engineering, University of Essex, UK, and the International Computer Institute, Ege University, Izmir, Turkey. Email: muge.sayit@essex.ac.uk, muge.sayit@ege.edu.tr.
This work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 121E033.

[1]https://www.sandvine.com/phenomena

Multipath communication between a video server and clients is a promising technique to provide higher throughput to streaming applications, which can be useful for meeting the requirements. Multipath TCP (MPTCP) as an extension of TCP, enables clients to open more than one subflow within the same TCP connection [1]. The redundancy offered by MPTCP utilizes the aggregation of bandwidth. Therefore, while this protocol has all the advantages of TCP, such as reliable transmission, flow, and congestion control, it provides an increase in throughput with the use of more than one subflow belonging to one particular MPTCP connection.

However, if the delay difference among the subflows is high, then goodput may be affected negatively due to the Head-of-Line (HoL) blocking problem [2]. Therefore, the path selection for the MPTCP subflows is an important problem that has gained attention. The subflow paths can be selected at the network layer by using Software Defined Networking (SDN) technology [3]. With SDN, the controller that is responsible for managing the SDN domain, can select a set of paths for the MPTCP subflows and assign the related forwarding rules to the switches by using a southbound protocol such as OpenFlow.

The performance of MPTCP is affected by the characteristics of the subflow paths on the network layer, such as bandwidth, delay, hop count, congestion level, and delay differences of the paths. Therefore, there are many proposals for the path selection on the network layer, where MPTCP subflows are transferred over those paths. QoE is mainly affected by the bandwidth of the paths. However, the delay difference of the paths is also another criterion that has a vital impact on QoE. In the tests that we conducted for our previous work [3], [4], we observed that if the delay difference between paths is over 100 ms, there is a 2-second duration of outage in 9 seconds of video, which corresponds to more than 20% of the total video duration. These observations align with those made in [5]. The duration of outage equals zero when there is no delay difference for the same bandwidth values. This observation shows that both bandwidth and delay differences should be considered in the path selection problem for the MPTCP subflows.

In this study, we focus on the path selection problem for enhancing the performance of video streaming applications. This work integrates emerging network technologies and protocols with one of the most widely used Internet applications. We explore a scenario in which HAS clients are connected to an SDN-based edge network, and a video server (i.e., a CDN server) is located within the network. Alternative scenarios are also possible. Our path selection approach is applicable within an SDN-based data center employing MPTCP [6], where a group of HAS clients connects to the servers of a video streaming or on-demand service.

The paths are selected by the SDN controller, which is responsible for the management of the domain. Paths are determined by a genetic algorithm and the path selection criteria which differs from the literature. The contributions of our work can be listed as follows:

- We select the MPTCP subflow paths on the network layer by jointly considering bandwidth and the delay differences among the paths. There is no study in the literature that jointly considers these criteria. We also take into account the disjointness of the paths.
- We propose a novel genetic algorithm to solve the problem which finds solutions within a small amount of time for complex optimization problems. Hence, the performance of the video streaming application does not deteriorate.
- We run experiments comparatively to evaluate the performance of the proposed approach. The experimental results not only show the performance of the proposed approach but also present how video streaming applications are affected by the MPTCP path characteristics.
- We also evaluate the performance of the proposed path selection algorithm, along with several other algorithms, by measuring their performance across different schedulers and buffer sizes.

The rest of the paper is organized as follows. Section II provides background studies on MPTCP path selection methods and the related work on genetic algorithms. Section III introduces the proposed solution for the path selection problem in detail. The results of the comparative performance evaluation are presented in section IV. We end the paper with a conclusion in section V, followed by the references.

## II. Background and Related Work

### A. MPTCP Characteristics and Path Selection for MPTCP Subflows

With MPTCP, the client and the server can communicate by using more than one subflow. This feature of MPTCP provides higher throughput thanks to the multiple path transmission, and higher connectivity (i) with the use of an effective scheduler, and/or (ii) with the subflows using different paths on the network layer.

*1) MPTCP Schedulers and Subflow Selection on the Transport Layer:* The MPTCP scheduler is responsible for selecting subflows for the packets that are being sent, to achieve efficient and fair resource utilization across multiple paths. This selection is based on factors such as available bandwidth, latency, and congestion. Numerous studies have proposed remarkable MPTCP schedulers. One of them is minRTT, which takes into account Round-Trip Time (RTT) when distributing packets. The MPTCP scheduler sends packets over the subflow with the lowest RTT until its congestion window is filled. Subsequently, packets are sent over the next faster subflow. While this approach maximizes throughput when using symmetric network paths, it suffers from HoL blocking when network paths are heterogeneous [7].

Several schedulers have been proposed to address the HoL blocking problem when MPTCP is running over heterogeneous paths. Among them, the Delay Aware Packet Scheduler (DAPS) scheduler [8] is designed to ensure that the data segments are received in the correct order. In addition, it utilizes the MPTCP receiving buffer by avoiding delay caused by congestion of the packets in the receiver buffer. DAPS mainly focuses on organizing how the packets are received rather than how fast they are transmitted.

In contrast, the Blocking-prediction Enabled Scheduling Technique (BLEST) scheduler proposed in [9] aims to reduce the risk of HoL-blocking by selectively disregarding some subflows during packet distribution. Another proposed scheduler is the Earliest Completion First (ECF) scheduler [5]. ECF estimates the completion time for each subflow based on factors such as data amount and transmission rate. It prioritizes the subflow with the earliest expected completion time, thereby reducing transfer time and enhancing responsiveness. In [10], a scheduler, called MP-DASH, considering DASH characteristics is proposed. The MP-DASH scheduler takes user interface preferences, video chunk size, and delivery deadlines as input and determines the best strategy for fetching video chunks over multipath connections. In conclusion, path selection at the network layer and MPTCP schedulers can collaborate to optimize the utilization of multiple paths, where the path selection approach determines the available paths and schedulers manage traffic distribution and transmission. For more information on MPTCP schedulers, a comprehensive survey can be found in [11].

Another influential factor on throughput at the transport layer is the network buffer size. The network buffer size on the client side plays a crucial role in managing the flow of data between the client and the network. The optimal buffer size depends on factors such as network conditions, traffic patterns, and system capabilities. In [12], the authors investigate the effect of buffer size on the performance of the original MPTCP and MPTCP with Constraint-Based (CP) scheduling, regarding both symmetric and asymmetric paths. Overall, the paper presents that CP scheduling, especially with the radical method, can improve the performance of MPTCP by effectively managing buffer limitations, preventing throughput degradation, and maintaining better RTT balance between paths. It provides insights into how different buffer sizes and scheduling strategies impact MPTCP's performance over multiple paths with varying characteristics.

In this study, we focus on the path selection problem at the network layer. Hence, our proposed approach can work with any proposals related to schedulers and various buffer sizes.

*2) Path Selection on the Network Layer for MPTCP subflows:* The enhancement of throughput provided by MPTCP does not only depend on the scheduler but also on the characteristics of the paths on the network layer, used by the subflows. Indeed, if the bandwidth available on the paths used by MPTCP subflows is notably restricted, the overall throughput remains constrained, regardless of the effectiveness of the scheduler employed at the transport layer. Consequently, numerous strategies have been introduced in the literature to address the path selection issue for MPTCP subflows, with a primary focus on the path selection problem on the network layer.

Several metrics can be considered for the path selection of MPTCP subflows. Hop count, bandwidth, congestion level, delay, path disjointness, and delay differences of the paths are among the factors considered in path selection. In [13], the path selection for MPTCP subflows is performed based on hop count. In [14], the authors propose to select as many as necessary paths starting from the shortest path so that the bandwidth of the selected paths exceeds the desired value. These studies haven't considered further metrics including path disjointness, delay difference, and path congestion.

The bandwidth values of the selected paths are one of the most important criteria that affect throughput. In [15], MPTCP subflow path selection is done based on the available bandwidth values of the paths. Bandwidth values of the paths are also considered in the selection of the paths for MPTCP subflows in a data center in the studies [16] and [17]. In [18], authors focused on the efficiency of MPTCP in optical networks, and the paths with the highest bandwidth are selected in their approach. The general approach used in bandwidth-based path selection is preferring paths with maximum available bandwidth values. In order to improve the performance of MPTCP transmission between the nodes in the Mobile AdHoc Networks (MANET), the signal quality between the nodes is considered in [19]. In some scenarios, HAS clients could be mobile users within a 4G/5G/5G+ network. In this case, the path characteristics on the last hop, i.e., in the wireless part, should also be considered. In [20], the overall performance of wireless networks is improved by tracking the MPTCP path capacity and selecting the most fitting paths to lower the number of out-of-order packets in an SDN network. This approach can be combined with our proposal by considering the characteristics of end-to-end paths.

Using more than one path in an MPTCP connection provides higher throughput due to the aggregation of bandwidth capacity of each path. However, the delay differences between the path pairs used by the MPTCP may cause the Head of Line (HoL) blocking problem, and as mentioned earlier, the performance of applications using MPTCP is affected negatively in terms of increased delay and decreased goodput [2], [4], [21]. Therefore, the delay differences between different subflow paths should be taken into account as well as path capacities in the path selection problem. The delay difference between the paths is not taken into account in any of these studies. However, it significantly affects the throughput, hence, it should be considered in the selection of the paths, especially for video streaming applications [22].

In [23], an algorithm is proposed to assign subflows requested by the MPTCP client, by selecting paths among fully disjoint paths. Then, the paths with limited bandwidth are eliminated from the selected paths. In [24], the authors propose an approach that selects the shortest and fully disjoint paths for MPTCP subflows in an SDN-based data center. In this study, the bandwidth and delay criteria are not considered. In [25], path selection is done by selecting the paths with maximum available bandwidth among fully disjoint paths. The SDN controller compares the delay of each path to the average delay and does not use the paths with high end-to-end delay. In [26], an optimization model is proposed to select fully disjoint paths and decide on the number of subflows by considering the delay difference of the paths. However, this study does not take into account the bandwidth of the paths. These studies consider important factors, such as bandwidth, delay differences, and path disjointness that affect QoE in video streaming applications, but assigning fully disjoint paths may not always be possible due to the topology constraints.

If there are not enough disjoint paths in the topology, then partially disjoint paths can be used for the MPTCP subflows. In our previous work [3], we proposed an approach for selecting paths for MPTCP subflows by using partially disjoint paths and by considering the bandwidth values of the paths. However, delay differences in the paths were not considered in the study.

To enhance the performance of video streaming applications, it is essential to consider both the bandwidth and delay differences of the paths utilized for MPTCP subflows. None of the mentioned studies above concurrently addresses these criteria. In this study, we present an approach for MPTCP subflow path selection that takes into account both bandwidth and delay differences. Moreover, the proposed algorithm selects partially disjoint paths, considering the congestion level of the shared links on these paths. As a result, there is no requirement for the paths to be completely disjoint in the topology.

In this work, we implemented our path selection approach in an SDN domain. Nevertheless, it's worth emphasizing that our approach remains applicable as long as there exists a technology permitting path selection at the network or link layer. Multi-Protocol Label Switching (MPLS) and optical networks can be given as such examples [26]. Another protocol that holds the potential for this purpose is the Locator/Identifier Separation Protocol (LISP). In [27], the selection of paths for MPTCP subflows is accomplished through the utilization of LISP. The study demonstrates the routing of subflows across diverse Internet Service Providers (ISPs).

## *B. Genetic Algorithm (GA) and GA Based Approaches for Multipath Transmission*

The genetic algorithm is a search algorithm that takes the principles of natural selection and genetics into account. This algorithm is capable of finding the most optimal or near-optimal solutions to difficult problems that can take a long time to be solved by other methods [28]. This algorithm is mostly used to solve optimization problems and uses a completely different method compared to brute force algorithms.

Initially, the algorithm creates a random population that contains a series of possible solutions to the problems. Each solution in the population is called a chromosome and each bit in a chromosome is a gene. To evaluate the accuracy of each solution, a fitness function is defined by taking into account problem characteristics. This function considers the solution of the problem and assesses the values of the chromosomes, which shows how good they are for solving the given problem.

To form the next generation of the population, particular parents are selected and combined which results in creating new solutions called *off-springs*. This process is called crossover. Parent selection is a very crucial process that leads to producing better and fitter off-springs. The final stage in the genetic algorithm is mutation. In this stage, several off-springs are randomly selected and the values of one or more of their genes are randomly changed based on a certain probability. The genetic algorithm is a powerful tool to solve large and complex problems. It has been used to find solutions for various networking problems such as network traffic prediction [29] and routing problems [30].

In [31], the genetic algorithm is utilized in multi-path heterogeneous wireless systems. The goal of the algorithm is to minimize the delay differences of MPTCP subflows by using a genetic algorithm-based congestion control algorithm. Therefore, the genetic algorithm is used on the client side, rather than for the path selection problem on the network, which is the focus of our study. The authors used the genetic algorithm for the placement of SDN devices to minimize the routing and deployment cost. They showed that a genetic algorithm-based approach is more scalable and powerful than a Mixed-Integer Linear Program (MILP) model in [32]. The selection of paths between peer-to-peer (P2P) overlay nodes is done by a genetic algorithm in [33]. In the study, the authors use P2P video characteristics within the genetic algorithm to select the optimal nodes in the overlay network. None of these studies focuses on the network path selection problem and uses network metrics such as available bandwidth and delay.

Several proposals use genetic algorithm-based network path selection for various purposes. In [34], the authors used the genetic algorithm to select the multiple paths in a data center. Their algorithm minimizes path length and maximizes link usage diversity to increase transmission rate. In [35], a genetic algorithm is developed to solve the Multi-constrained routing (MCR) problem which is NP-complete. The main objective of MCR is to determine the feasible path in the network that satisfies transmission delay and transmission success ratio constraints. A genetic algorithm-based approach is proposed for selecting the shortest path in terms of delay between the source and destination in [36]. None of these studies addresses the path selection problem for multiple path transmission by considering the delay difference and bandwidth of the paths. Hence, these solutions do not apply to the problem we focus on in this study.

## III. Proposed Approach

In this section, we give the details of the proposed genetic algorithm-based path selection and explain the framework that is used in this study.

### A. Implementation of MPTCP Path Selection in an SDN Domain

In this study, the MPTCP-enabled server and clients reside in an SDN domain. The selection of the MPTCP subflow paths is done by the SDN controller which manages the domain. For this purpose, it periodically measures traffic statistics and calculates the available bandwidth values of each path. It also keeps track of the clients in its domain. An illustration of the system is given in Fig. 1. The figure illustrates a scenario in which the client has three subflows. The network paths assigned by the controller are disjoint in the first and second hops from the client side, while two of them utilize a common link on the third hop. Additionally, the figure illustrates the application of the genetic algorithm, running as a third-party application.

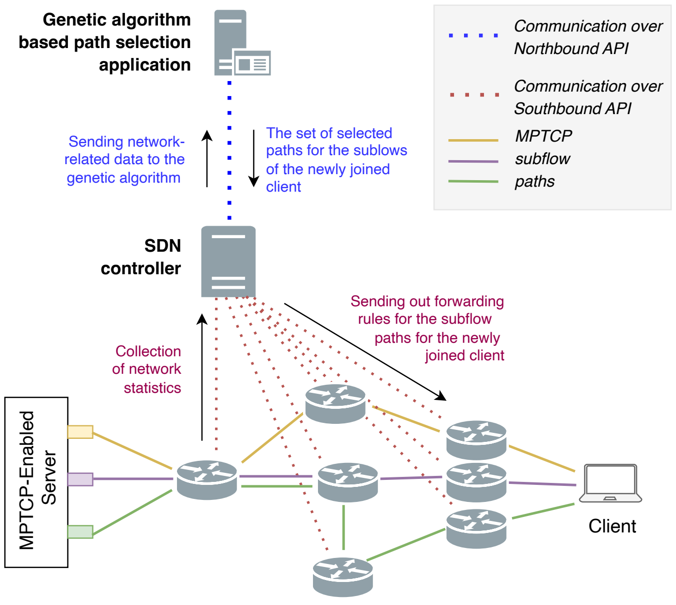

Genetic algorithm based path selection application
Communication over Northbound API
Communication over Southbound API
MPTCP subflow paths
Sending network-related data to the genetic algorithm
The set of selected paths for the sublows of the newly joined client
SDN controller
Sending out forwarding rules for the subflow paths for the newly joined client
Collection of network statistics
MPTCP-Enabled Server
Client

Fig. 1: Illustration of the System

When a client starts the video streaming application, it establishes an MPTCP connection with the server. Suppose the clients have $N$ IP numbers, where $N \geq 1$. This can either be provided with multiple physical or virtual interfaces. We assume the network operator and video streaming company are in cooperation and the controller knows the number of $N$.

When the controller gets the first packet of the connection from the first-hop switch, it detects that the client is initiating an MPTCP connection by examining the TCP header and the MP_CAPABLE option. Subsequently, the controller runs the path selection algorithm for selecting up to $N$ paths for the subflows of the newly joined client. At this stage, the controller only has information about the first subflow *(IP, port number)* pair. Therefore, it only sends forwarding rules about the path of the first subflow.

During the initial handshake process of MPTCP connection establishment, both the client and the server generate a 64-bit random key. These keys are added to the MP_CAPABLE option. A new key is generated by utilizing the initially generated keys, serving the purpose of authentication for subsequent subflows [37].

To establish an additional subflow, a new SYN exchange is performed, which includes the IP address and port that will be used. During this stage, the MP_JOIN option, containing the Message Authentication Code (MAC) of the key generated during the connection establishment, is added to the SYN packet [38]. The first-hop switch to which the client is connected sends each packet with the MP_JOIN option to the controller. This is because there is no flow information in the forwarding table of the switch related to this *(IP, port number)* pair. The controller can detect which MPTCP connection the additional subflow belongs to. Consequently, it can utilize the previously determined path information for each subflow of the client without the need to rerun the genetic algorithm. This mechanism ensures that the genetic algorithm is executed only once to determine the paths for all subflows of a client.

This process continues until $N$ requests are sent to the server to establish all subflows. The first hop switch sends a packet for each time it gets an MPTCP subflow establishment packet from the client. Hence, the controller has the information about *(IP, port number)* for other subflows and sends the forwarding rules related to the paths of those subflows. The genetic algorithm that is used for the selection of the paths may select less than $N$ subflow for the client. In that case, the controller does not send forwarding rules for non-selected subflows. Hence, the client cannot establish connections for those subflows and the number of its subflows is limited. If the selected path becomes suboptimal, the genetic algorithm can be rerun.

The SDN controller continuously monitors the network state, which involves collecting statistics and status information from network devices. The Southbound APIs, like OpenFlow, are used for this purpose. These APIs allow the controller to ask devices for their status and other relevant data. The controller also receives real-time updates regarding link failure or newly added network devices. Therefore, the controller has an accurate and up-to-date view of the network. The controller transfers network-related information to the genetic algorithm application through its northbound API.

In this study, the genetic algorithm is rerun each time a new client joins the system. Once the genetic algorithm determines the optimal paths, the controller updates the routing tables of the network devices using the southbound APIs, which are associated with the flow rules defined for the newly joined client. While routing paths for existing clients are not rerouted in this study, it is possible to do so by triggering the execution of the genetic algorithm. In this scenario, the genetic algorithm can be executed to determine new paths for a subset of clients, selected based on a threshold for changes in aggregated bandwidth.

In the next sections, we define the path selection problem as an optimization model and then provide our proposed solution which is based on a genetic algorithm.

### B. On the MPTCP Path Selection Problem

We define the MPTCP path selection problem as a multi-objective optimization model that considers the aggregated bandwidth of the paths and the delay differences between them. The optimization model is given in formulas 1-6. In the model, $C$ and $P$ represent the set of clients and the set of paths in the network topology, respectively. $x_{c,p}$ is the binary variable that shows if client $c$ uses path $p$. The first objective equation aims to minimize the delay differences ($dd$) of each pair of the selected paths for a client in the system. The second objective equation maximizes the aggregated available bandwidth value ($bw$) of the selected paths. In the equation, $abw_p$ represents the available bandwidth of path $p$. The paths chosen for each subflow are partially disjoint, which means if the links of the path chosen for the initial subflow have a bandwidth capacity of more than the predefined value, they can be used in the other subflows as well. If the paths are partially disjoint, then the congestion level of the common links of different paths should be less than a threshold. Equation 3 provides this constraint, where $cong\_level$ represents the maximum congestion level that is measured in the common links of the paths and $thr_{cong}$ represents the threshold. Even if the delay difference between two paths is small, paths with delays higher than an acceptable value are not selected. This is provided by the 4th constraint, which imposes a threshold, $thr_{delay}$, to limit the delay of the paths. Constraint 5 limits the selected number of paths so that this number is less than the total number of different IPs used by the client.

$$\min_{dd} \sum_{c\in C} \sum_{p=1}^{|P|-1} \sum_{r=p+1}^{|P|} x_{c,p}.x_{c,r}.|d_p - d_r| \tag{1}$$

$$\max_{bw} \sum_{c\in C} \sum_{p=1}^{|P|} x_{c,p}.\frac{abw_p}{\sum_{h\in C} x_{h,p}} \tag{2}$$

subject to:

$$x_{c,p}.cong\_level \leq thr_{cong}, \quad \forall p \in P, \forall c \in C \tag{3}$$

$$d_p, d_r \leq thr_{delay} \;\; \forall p, r \in P \tag{4}$$

$$\sum_{c\in C} x_{c,p} \leq N \tag{5}$$

$$x_{c,p} \in \{0, 1\} \tag{6}$$

We note that optimization models or multiple constraint optimal path selection approaches have been proposed earlier for multipath communication [3], [26], [39]. In none of these studies, bandwidth, delay differences, and congested links are jointly considered in the models and the provided solutions. Using all of these factors increases the computational complexity of the models. The optimization model given in the formulas between 1-6 includes binary variables. The optimal path assignment for the clients is an instance of the generalized bin packing problem [40], which is an NP-complete problem. Therefore, the solution of the model is not suitable for using it in our system due to the time limitation. In our system, the SDN controller should assign paths to the clients after they start a video streaming application. To find a suitable set of MPTCP paths for the clients, we utilize a genetic algorithm. Genetic algorithm-based solutions can provide a good balance in the computational time and solution quality trade-off when the problem is NP-hard [32]. The details of the proposed genetic algorithm-based solution are given in the next section.

## *C. Genetic Algorithm-based Path Selection*

In the environment established in this study, as the client joins the network, the communication between the client and the controller starts by exchanging packets. The client initiates the MPTCP connection by sending a SYN packet including the MP_CAPABLE option.

When the Floodlight controller receives the initial packet, it executes a genetic algorithm to determine the paths for the subflows of the client. As described in the previous chapter, the first path is assigned for the first subflow of a client upon receiving a TCP_SYN message.

Each client is designed to have three MPTCP subflows in this study. However, the genetic algorithm can result in choosing one, two, or three subflows for the clients. Although the client sends a SYN packet to the controller requesting three subflows, the controller can decide to assign only one, two, or all three subflows based on the outcome of the genetic algorithm. Based on the result received by the genetic algorithm, the controller establishes the paths and the client starts sending data packets.

As stated previously, the genetic algorithm must be designed so that the MPTCP subflows' paths with the highest bandwidth and the lowest delay difference would be selected. Also, if there exist common links among the selected paths, then the congestion level of those common links should be considered. We have developed a genetic algorithm that selects the paths with these characteristics by considering the constraints given in the optimization model in the previous section.

The main steps of the genetic algorithm are given in Algorithm 1. The algorithm starts by creating an initial population of chromosomes. In this study, a chromosome is defined as a $1xm$ sized vector, where $m$ is the maximum number of the subflows. This number depends on the number of physical or virtual interfaces of the client. The elements of the vector are the IDs of the paths. $i^{th}$ element of the vector shows the selected path ID for the $i^{th}$ subflow. The number of subflows opened for an MPTCP connection can be equal to or less than $m$. The element of the vector is set to 0 for the unused subflows.

The SDN controller has a list of all possible paths between the client and the server. Before the generation of the initial population, the controller sorts all paths regarding their delay values. Paths with delay values exceeding the threshold are eliminated and cannot be included as members of the population. After this process, the controller generates the initial population by selecting the chromosomes randomly. Therefore, the population is the subset of all possible path set combinations that can be selected for the MPTCP subflows of the new client. The generation of the initial population is given in Algorithm 2. The algorithm creates chromosomes until the population reaches the population size. In the $5^{th}$ line of the algorithm, a new chromosome is created and the common links of the paths of the chromosome are checked whether the congestion level is less than the predefined threshold in the $6^{th}$ line. If this condition is held, the new chromosome is added to the initial population.

After the population is set, the fitness values of the chromosomes in the current population are calculated. The algorithm used for fitness value calculation is given in Algorithm 3. The fitness of each chromosome is calculated by using equation 7. In the equation, $\zeta(.)$ and $\xi(.)$ functions are given in formulas 8 and 9, respectively, which are modified versions of the equations given in 1 and 2. In the formulas, $P_c$ represents the set of paths that are assigned to the client and $H_p$ represents the set of clients whose packets are transferred over the path $p$. To align the ranges of bandwidth and delay values, the values calculated by these functions are first normalized between 0 and 1 before they are given as the inputs for equation 7. The fitness value is set to 0 if the congestion level of the common links of the selected paths is higher than the threshold, which

**Algorithm 1:** Genetic Algorithm Steps

```
Input: $M_i$: $i^{th}$ parent
       $O_i$ : $i^{th}$ offspring
       $n$: Population size
       $P'$ : the set of temporary population
       $P_{new}$ : the generated new population
       $f_p[1..n]$: an array whose elements are fitness values of the chromosomes
       $\mu$ : Mutation probability
1  $P_{new}$ = $generate\_initial\_population(n)$;
2  while no of generations $\leq MaxGenerationCount$ do
3      $P_{current} \leftarrow P_{new}$
4      $f_p, P_{new} \leftarrow calculate\_fitness(P_{current})$
5      // Tournament selection:
6      while $|P'| \leq n$ do
7          $P' \leftarrow \emptyset$
8          $M_i, M_j$ = Randomly select two chromosomes in $P_{current}$
9          if $f_p[M_i] \geq f_p[M_j]$ then
10             $P' = P' \cup M_i$
11         end
12         else if $f_p[M_i] < f_p[M_j]$ then
13             $P' = P' \cup M_j$
14         end
15     end
16     // Two-point crossover:
17     while $|P_{new}| \leq n$ do
18         foreach $M_i \in P'$ & $M_j \in P'$ do
19             $O_i$ , $O_j$ = twoPointCrossover($M_i$, $M_j$);
20             $P_{new} = P_{new} \cup \{O_i, O_j\}$
21         end
22     end
23     // Mutation:
24     foreach $O_i \in P_{new}$ do
25         $O_i$ = Mutate ($O_i$) with probability $\mu$;
26     end
27     no of generations++;
28 end
```

**Algorithm 2:** Generation of the Initial Population: $generate\_initial\_population$

```
Input: $n$: Population size
       $m$: max no of MPTCP subflows
Output: $P_{new}$: new population
1  $P_{new} \leftarrow \emptyset$
2  $inPopCount = 0$
3  while $inPopCount < n$ do
4      //a new chromosome is created
5      $c \leftarrow$ randomly select paths up to $m$
6      if checkCongestionLevelonCommonLinks(c) then
7          $P_{new} \leftarrow P_{new} \cup \{c\}$
8          $inPopCount + +$
9      end
10 end
11 Return $P_{new}$;
```

**Algorithm 3:** Calculation of fitness values and selection of elitist members: $calculate\_fitness$

```
Input: $P_{current}$
Output: $f_p[1..n]$: fitness values of the chromosomes in $P_{current}$
        $P_{new}$ : next population
Data: $\alpha$, $\beta$ : normalization values
      $n_e$: no of elitist members
      $P^*$ : set of elite members
1  $n = |P_{current}|$ \\population size
2  $P_{new} \leftarrow \emptyset$
3  foreach $i \in P_{current}$ do
4      $f_p[i]= \alpha.\zeta(.) + \beta.\xi(.)$;
5      if checkCongestionLevelonCommonLinks(i) then
6          $f_p[i]= 0$;
7      end
8  end
9  $P_{sorted}$ = Sort members $\in P_{current}$ based on fitness values
10 $P^* \leftarrow$ First $n_e$ members $\in P_{sorted}$
11 $P_{new} \leftarrow P_{new} \cup P^*$;
12 Return $f_p, P_{new}$;
```

is done between the 5th and 7th lines of Algorithm 3. The algorithm does not only calculate the fitness values of the chromosomes in the current population, it also selects the best chromosomes based on the fitness value and directly adds these best chromosomes to the next generation. This set of chromosomes is called elitist members. The number of elitist members is a parameter that is predefined.

$$\alpha.\zeta(.) + \beta.\xi(.) \tag{7}$$

$$\sum_{p=1}^{|P_c|-1} \sum_{r=p+1}^{|P_c|} |d_p - d_r| \tag{8}$$

$$\sum_{p=1}^{|P_c|} \frac{abw_p}{\sum_{h\in H_p} +1} \tag{9}$$

Algorithm 3 calculates the fitness values of the chromosomes in the current population and returns to the next population that is initially filled with elitist members.

The next generation construction process continues with the *Tournament selection* method in Algorithm 1. In this method, a subset of the chromosomes is selected randomly, and the members of that set compete with each other by comparing the fitness values. This method is run between the 6th and 14th lines in the algorithm. The ones with better fitness values will become parents of the next generation.

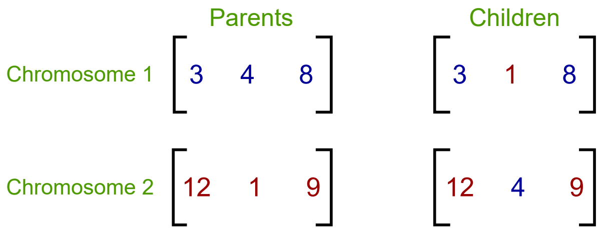


Fig. 2: Two Point Crossover Method

After selecting the parents with the *Tournament selection* method, a crossover operation is implemented over the selected pair of chromosomes. For crossover operation, the *Two-point crossover* method is applied. In this method, two points are selected from the parent chromosomes and the genes in between these two points are swapped. This operation is shown in Fig. 2. In the scenario given in the figure, the second genes of both parents are swapped and two different offspring, i.e. children, are obtained. Two-point crossover method is run between the 16th and 22nd lines in Algorithm 1.

Offspring 1
3 1 8
Offspring 1 after Mutation
3 0 8

Fig. 3: Mutation Operation

The next stage is mutation. Finding good chromosomes is the most important purpose in the whole process and the mutation operation helps expand the search area by generating new chromosomes. At this stage, the genes of the chromosomes are modified with a probability value. In the scenario given in Fig. 3, the second gene of the first offspring is mutated. The fitness function values are calculated for the mutated chromosomes. At the end of this stage, the chromosomes having the highest fitness values are selected as the members of the new generation.

The genetic algorithm continues to run and executes these stages in each round until it reaches the generation number limit.

The entire framework developed for this study has been shared for public access on Zenodo.[1] All software developed and utilized in this study, including the genetic algorithm code, tools for constructing the SDN environment and Industrial DASH, and Mininet Python scripts, are available on the project's Zenodo page.

## IV. Performance Evaluation

In this section, we provide the performance evaluation of the proposed approach. First, we describe the emulation setup and then analyze the evaluation results.

### A. Evaluation Setup

*1) Path Selection Algorithms:* To evaluate the performance, we have conducted various experiments over the Mininet environment [41]. Mininet is an emulation platform that provides the infrastructure to emulate topologies and physical network components and to implement SDN-based approaches. The Floodlight SDN controller[2], which is an open-source controller, was deployed to manage the network and provides routing of the packets based on rules of the selected paths by programming OpenFlow-enabled switches.

A HAS application is selected as the video streaming application to evaluate the performance. Industry DASH has been used as the HAS client to stream the video[3]. The server sends Big Buck Bunny as the video content which has four

[1]https://zenodo.org/records/11205757
[2]https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview
[3]https://github.com/Dash-Industry-Forum/dash.js

different qualities with an average bitrate of 1500 Kbps, 2000 Kbps, 2500 Kbps, and 2600 Kbps. The duration of the video is 9 minutes.

The experiments are conducted over three different topologies. The first topology is from the Internet Topology Zoo [4], in which real-world topologies can be found. The CompuServe network topology is mainly used by ISPs. This topology consists of 11 switches representing the states of the United States of America and 14 links to connect them. As the second topology, we used the topology given in [25], which is also implemented as a comparison study. This custom topology contains 12 switches and 19 links to connect them. As the last topology, we deployed a well-known data-center topology, Jellyfish [42]. This topology is a combination of 20 switches with four degrees, which are connected randomly. In the topologies, all switches are OpenFlow-enabled and connected to the SDN controller. Servers and clients in these setups have three interfaces for Internet connectivity. In the real world, these connections can be wired or wireless, utilizing LTE and WiFi, or employing dual-stacked WiFi supporting IPv4/IPv6. Additionally, multiple SIM cards on a mobile phone can facilitate connectivity. Furthermore, opening more than one MPTCP connection is achievable by utilizing virtualized interfaces within a virtual machine.

In the conducted experiments, four sets of bandwidth values were employed across three distinct topologies to evaluate the performance of path selection algorithms under varying conditions. It is noteworthy that bandwidth values underwent dynamic fluctuations throughout the experiments due to the inherent dynamism of DASH traffic and the presence of competing flows on shared links. In the initial test scenario, the bandwidth values of the paths are theoretically enough to send the video with the highest bitrate to each client. Five MPTCP clients joined the network every 4 seconds. In the second test scenario, the number of MPTCP clients was increased to ten, with a slight reduction in path bandwidth, thereby prompting clients to contend for packets transmitting the video with the highest bitrate. The third test environment accommodated 20 MPTCP clients, alongside intentionally constrained bandwidth values on links to foster network congestion and heterogeneity. Finally, in the fourth test scenario, the number of MPTCP clients increased to 30 while bandwidth values were drastically limited, with a larger standard deviation. This was done to observe the path selection algorithms' behavior in a significantly congested and heterogeneous network. The average of the bandwidth value of the CompuServe, Custom, and Jellyfish for all client groups is given in Table I. Note that, these values represent the bandwidth of the links in the topologies. The bandwidth of end-to-end paths for subflows may deviate based on the bandwidth of selected paths. The Standard Deviation of the links of all three topologies and the client groups are given in Table II. In all test environments, the delay value of the links varies between 10 ms to 100 ms. Both delay and bandwidth values are distributed using a Poisson distribution.

To provide comparable performance results, four different path selection approaches were implemented, and experiments

[4]http://www.topology-zoo.org/dataset.html

TABLE I: Average Bandwidth Values of the Links in Three Topologies (in Mbps)

| Average | CompuServe | Custom | JellyFish |
|---|---|---|---|
| 5 Clients | 7.33 | 6.7 | 6.91 |
| 10 Clients | 13.06 | 14.09 | 13.7 |
| 20 Clients | 21.64 | 20.38 | 19.38 |
| 30 Clients | 23.21 | 20.12 | 21.16 |

TABLE II: Standard Deviation Values of the Link Bandwidth in Three Topologies

| Average | CompuServe | Custom | JellyFish |
|---|---|---|---|
| 5 Clients | 2.46 | 2.5 | 2.16 |
| 10 Clients | 6.42 | 6.12 | 5.98 |
| 20 Clients | 7.76 | 6.87 | 7.77 |
| 30 Clients | 12.99 | 10.35 | 13.03 |

were conducted over the same network topologies and conditions. Since three path selection features were considered in the proposed approach, studies considering a combination of those features were implemented for comparison:

- ***Partially Disjoint Bandwidth-based***: As proposed in [15], the algorithm assigns paths to a client by taking into account the maximum available bandwidth in this study. Another goal addressed in this approach is the use of partially disjoint paths, which is also applied in the proposed genetic algorithm. Once a path is assigned to one of the subflows of a client, the other subflows do not use the links of the same path unless the bandwidth capacity of the links of the selected path is greater than a predefined value.
- ***Partially Disjoint Delay-based***: In this comparison study [14], the algorithm calculates the total end-to-end delay of all paths. When a client connects to the network, the algorithm assigns a path with the lowest latency. The algorithm uses links from the initially selected path for the other paths only if the links have a latency lower than the predefined value.
- ***Fully Disjoint Bandwidth-based***: In [25], the algorithm selects paths for the client subflows based on the maximum available bandwidth. However, to select paths, it uses a fully disjoint strategy. Consequently, once the algorithm specifies a path for a particular client, links used in that path cannot be used for other subflows of that client. Since this approach prioritizes the bandwidth, paths with high bandwidth but also high delay could lead to performance degradation. To overcome this issue, the algorithm determines the paths with high delay values by comparing the delay of each path with the average delay. The paths with high values are then deleted from the list of available paths to be assigned.
- ***Fully Disjoint Delay-based***: Delay differences of the paths are taken into account in this comparison study [26]. The main purpose of this approach is to choose fully disjoint paths while minimizing the end-to-end delay difference. The major concern with this approach is choosing paths with high end-to-end delays, as

only the difference in latency of the paths is considered. To avoid this situation, an upper bound is assumed for the end-to-end delay of the paths.

In the experiments, the mutation rate is set to 0.25 in the genetic algorithm. The number of generations is 30.

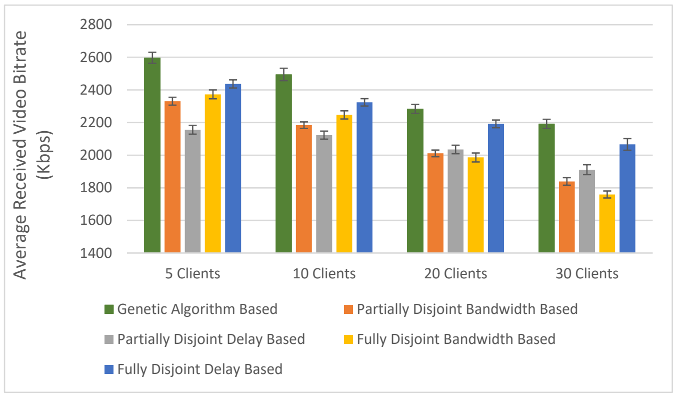
*2) MPTCP Scheduler and Buffer Size:* There are additional criteria that have an impact on QoE, such as the type of scheduler and the buffer size value on the client side. Although these criteria fall outside the scope of this study, we have conducted several tests to enhance the comprehensiveness of the research. The primary goal is to illustrate that the performance enhancement provided by the proposed path selection algorithm remains consistent regardless of the setting on the client side. To validate this claim, we have performed tests with three different buffer sizes as lower, default, and higher values which are 125 KB, 256 KB, and 512 KB respectively. The default MPTCP scheduler, implemented by using Linux kernel version 5.15.0-105-generic in the Ubuntu 20.04 operating system is based on RTT measurements of the subflows and is called minRTT. It schedules the segment to be sent over the subflow with the lowest RTT. The subflow having the lowest RTT can change over time due to the change of available spaces in their congestion windows. The other scheduler used in the experiments is DAPS, which, as mentioned in the previous sections, aims to maximize in-order segment reception at the destination. It focuses on the efficient use of the MPTCP receive buffer to minimize HoL blocking, rather than reducing latency.

## B. Evaluation Results

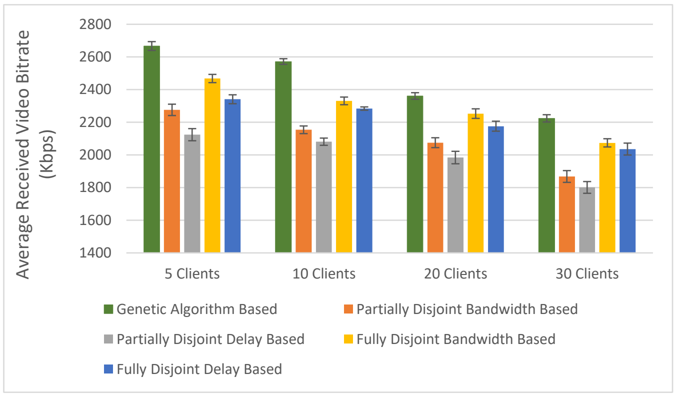
*1) Experiments with default scheduler and buffer size:* To evaluate the performance of each approach, the QoE metrics that are collected from each experiment are the average received bitrate, the average number of quality switches, and the average duration of the outages. The average received bitrate is the average bitrate of the video played by the clients. The number of quality switches is calculated by summing the quality differences between consecutive requests. Experiments were repeated ten times for each network configuration. Averaged values are presented in the graphs, where the confidence interval is 0.95. The minRTT scheduler was used as the default scheduler while the default value of the buffer size was 256 KB. To highlight the impact of path selection on throughput, we conducted tests using MPTCP without implementing any path selection mechanism at first.

In these initial experiments, tests were executed with 5 clients on the CompuServe topology, with paths having the average bandwidth and standard deviation given in Table I and II. In the experiments, the clients played the video with the lowest quality, where the average received bitrate was 1728 Kbps. The average number of video quality switches experienced by the clients was 7, while clients experienced 4 seconds of outage duration on average. Subsequent tests employing various path selection approaches highlight the significance of an appropriate path selection algorithm in enhancing received throughput.

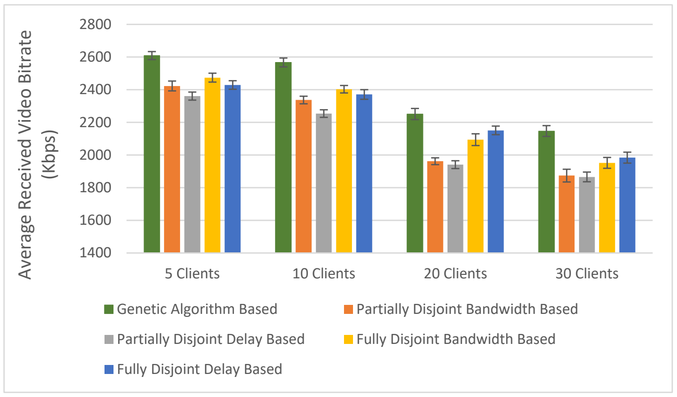
After observing the performance of MPTCP without using any path selection on the network layer, we conducted tests with various path selection algorithms and multiple clients. Fig. 4a, 4b, and 4c show the average received video bitrate of five, ten, twenty, and thirty clients where the network conditions change in each setting as previously described.



(a) CompuServe Topology



(b) Custom Topology



(c) Jellyfish Topology

Fig. 4: Average Received Video Bitrate

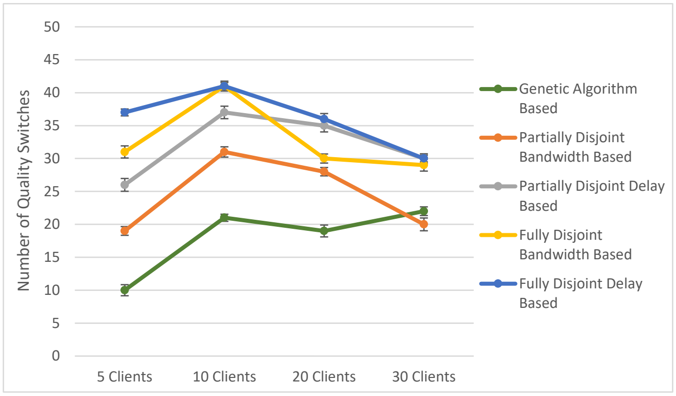
As depicted in the figures, our proposed approach consistently provides clients with higher bitrates across all scenarios. The clients utilizing our approach can play the video at a moderate rate even under the most adverse network conditions among the tested scenarios. Upon closer examination of performance, it becomes evident that path selection approaches considering delay values or differences are less adversely impacted by deteriorating network conditions, particularly notable in the CompuServe and Jellyfish topologies. Another observation that we got from these experiments is that the path approaches have the second-best performance in terms of received bitrate changes as the topology and/or the number of clients change.

As the number of clients increases, network conditions become increasingly constrained, posing challenges in delivering video packets. Despite variations in the performance of different path selection algorithms arising from diverse network conditions and topology characteristics, the genetic algorithm consistently outperforms other approaches across all client sets and topologies. Clients utilizing the genetic algorithm-based path selection stream the video at rates ranging from 2200 Kbps to 2600 Kbps. *This observation highlights the importance of considering all three path selection criteria to obtain higher quality in terms of received video bitrate*.

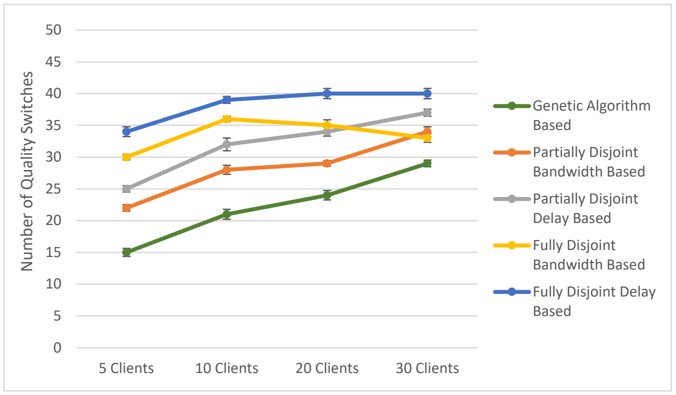
Fig. 5a, 5b, and 5c represents the average number of quality switches observed in the experiments done with different numbers of clients. The figures clearly illustrate that the clients experience a lower number of quality changes with the proposed approach using the genetic algorithm in comparison to other approaches.

The clients adjust the quality to either reduce the duration of outages or to enhance video quality when sufficient data is available in their buffer. Across all three topologies, clients utilizing fully disjoint bandwidth-based and fully disjoint delay-based approaches experience more than 35 quality switches to minimize outage duration or enhance quality. Meanwhile, the clients using the genetic algorithm-based approach experience 29 quality switches at most in all topologies. This demonstrates that clients can decrease the duration of outages and optimize video quality with fewer quality switches, thanks to our path selection approach. In the CompuServe and Jellyfish topologies, we note a declining trend for most applications as the number of clients increases. This trend indicates that clients predominantly stream video content at lower quality levels, with fewer attempts to enhance quality slightly. In other words, in congested networks where the number of quality switches is reduced, clients may become stuck at the lowest quality level. However, this pattern does not hold for our approach. As evidenced by the bitrate graphs, our approach enables clients to stream video content at higher quality levels despite experiencing fewer quality switches.

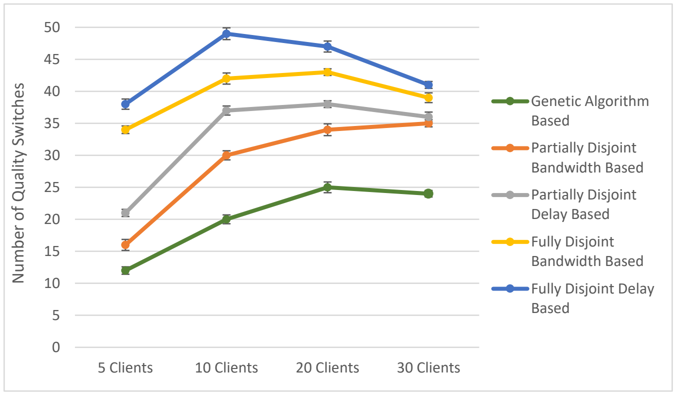
The duration of outages observed for all client sets are given in Figures 6a, 6b, and 6c. The duration of outage is a critical factor that affects user experience and primarily impacts perceived quality negatively. Our proposed genetic algorithm approach consistently outperformed the other path selection approaches in all three network topologies. The duration of outages observed in comparative studies has increased up to four times the value observed in the proposed approach in some scenarios as seen in the test results conducted on the CompuServe topology, where there were 5 clients. Conversely, the performance of comparative approaches varies across different topologies. Clients utilizing the partially disjoint bandwidth-based approach encounter the longest duration of outages in CompuServe and Jellyfish topologies, whereas



(a) CompuServe Topology



(b) Custom Topology

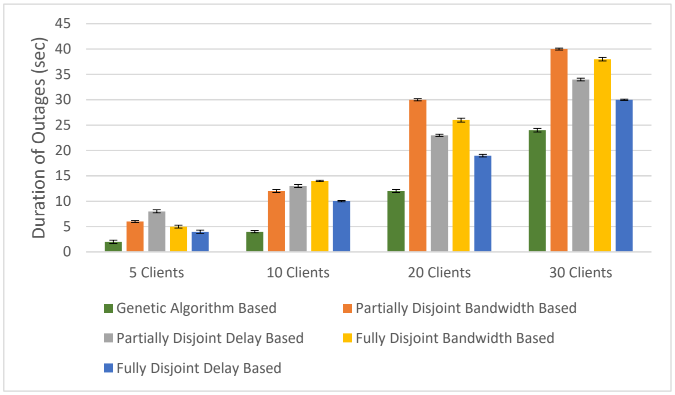


(c) Jellyfish Topology

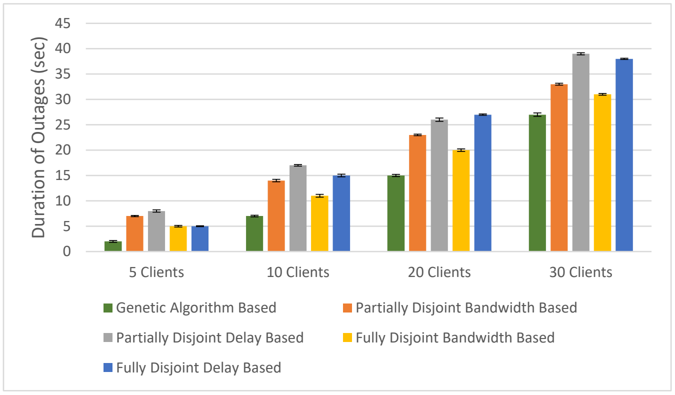
Fig. 5: Average Number of Quality Switches

those employing the partially disjoint delay-based approach experience the longest outages in the Custom topology.

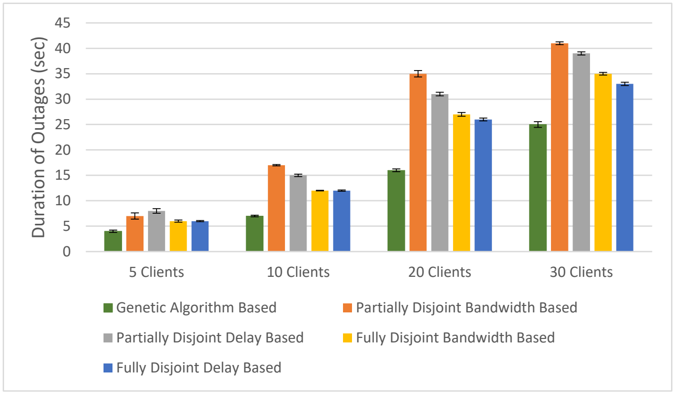
The main reason is that these two approaches consider one parameter solely, bandwidth or delay, and this might cause the selected paths to be good in terms of bandwidth (delay) and bad in terms of delay (bandwidth). The fully disjoint delay-based approach does not consider the bandwidth values of the paths. However, this approach focuses on delay differences among the paths rather than end-to-end delay values of the paths. In addition, the fully disjoint bandwidth-based approach eliminates the paths with a delay higher than a threshold. Even



(a) CompuServe Topology



(b) Custom Topology



(c) Jellyfish Topology

Fig. 6: Average Duration of Outages

this simple approach helps reduce the duration of outages. The fully disjoint bandwidth-based approach has performed better in terms of the duration of outages where the network conditions are good. However, the limitation in the bandwidth negatively affects the performance of this approach due to the restriction on the use of common links. *These results verify that (i) delay differences have more impact compared to delay values of the paths, (ii) Considering more than one parameter has a positive impact on minimization of the duration of outages, (iii) if common links are not used in the set of paths, the outage of duration might increase when the bandwidth is limited, due to the inefficient utilization of the resources.*

We also conducted tests considering a scenario where the bandwidth increases as the number of clients in the system increases. The obtained results were similar to those observed in the scenario with 5 clients. We omitted the graphs related to these experiments here to increase readability. However, interested readers can access the results of the related experiments on the Zenodo page of the study. [5]

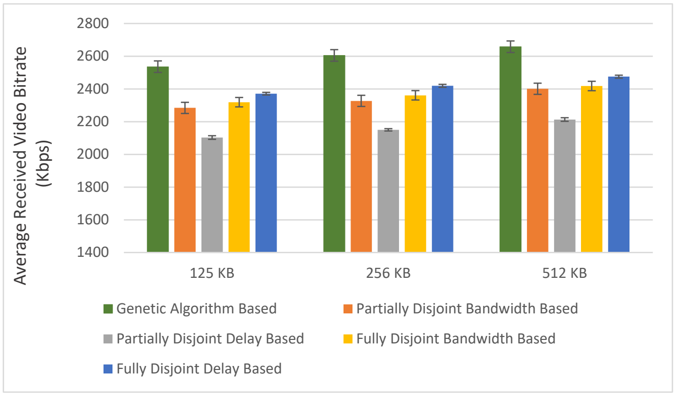
2) *Experiments with Different Schedulers and Buffer Sizes:* Another set of experiments was conducted to assess the impact of different schedulers and buffer sizes on the performance of the path selection algorithms. The objective of this section is to provide a more insightful conclusion regarding the performance of the proposed path selection algorithm. We aim to determine whether the achieved performance gain can be replicated through the utilization of an improved scheduler. Consequently, we can gain a deeper understanding of the contributions of the path selection algorithms at the network layer.

To accomplish this, we conducted experiments, comparing results with both the DAPS scheduler and the default scheduler. The experiments were carried out using buffer sizes set to 125 KB, 256 KB, and 512 KB within the CompuServe topology. The bandwidth values are distributed so that if two paths are assigned to a client, it enables the client to download the video at approximately the highest quality available.

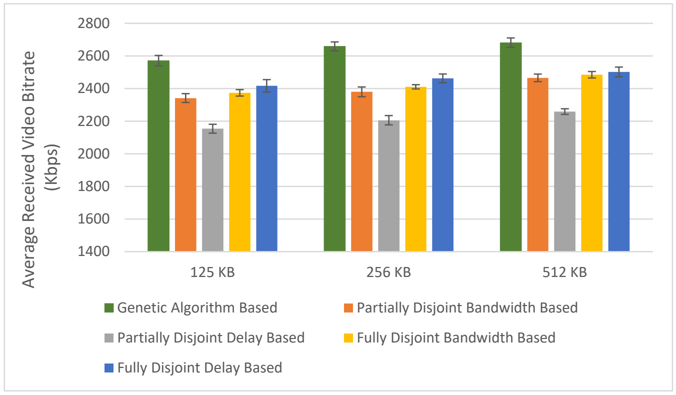
Fig. 7a and 7b show the average received video bitrate for 30 clients in the CompuServe topology. In the given network setting, the average received video bitrate with the DAPS scheduler is slightly better than with the default scheduler. The positive impact of the DAPS scheduler on the bitrates is similar across different path selection algorithms. DAPS effectively addresses the HoL blocking problem compared to the default scheduler. However, the average received video bitrate alone does not fully capture the significance of this positive impact. The reason is the greedy nature of HAS client software that leads the clients to download the highest possible quality under throughput constraints on the client side. Therefore, clients maximize video quality with both schedulers, with slightly higher values observed with DAPS due to its throughput improvement.

Fig. 8a, 8b represent the average number of quality switches observed at the clients with the given setting. The number of quality switches is a QoE parameter, which is primarily influenced by changes in network conditions. Our observations reveal that the DAPS scheduler significantly reduces these switches by prioritizing in-order reception, particularly benefiting path selection algorithms focused solely on delay, such as the partially disjoint delay-based approach. On average, we observed a 27.6% decrease in quality switches with this approach, compared to a 12% decrease with other strategies. This result highlights the persistence of HoL blocking issues, even when paths with low delay are selected. The lowest quality switching values are observed with the partially disjoint bandwidth based on the default scheduler. This can be attributed to its overall limited performance in terms of video

[5]https://zenodo.org/records/11209566

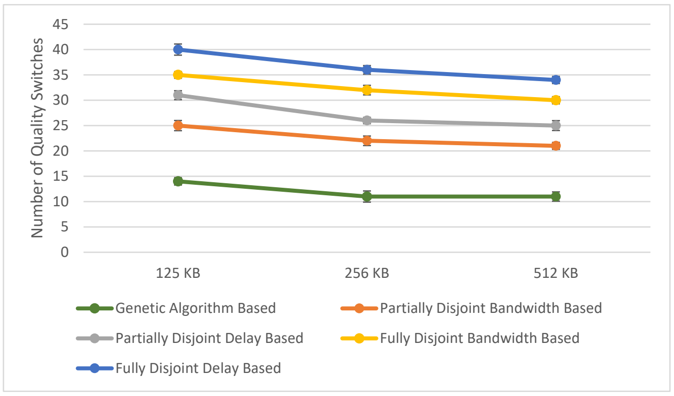


(a) CompuServe Topology, Default Scheduler

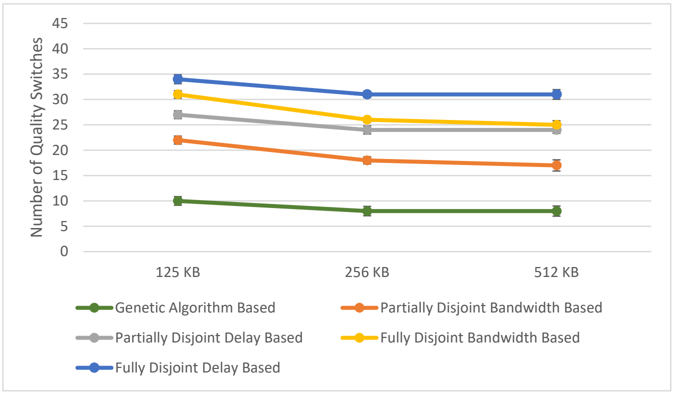


(b) CompuServe Topology, DAPS Scheduler

Fig. 7: Average Received Video Bitrate



(a) CompuServe Topology, Default Scheduler



(b) CompuServe Topology, DAPS Scheduler

Fig. 8: Average Number of Quality Switches

quality and duration of outages, resulting in fewer quality switches due to consistently low-quality video delivery.

The average duration of outage for the clients for all network settings is given in Fig. 9a, 9b. Because the DAPS scheduler focuses on the in-order reception of segments, its effectiveness is enhanced when coupled with path selection algorithms that consider delay values. We observed a significant performance improvement in partially disjoint delay-based and fully disjoint delay-based approaches, with an average decrease of 11%. On the other hand, the use of the DAPS schedule does not improve the performance in terms of the duration of outage for the bandwidth-based applications.

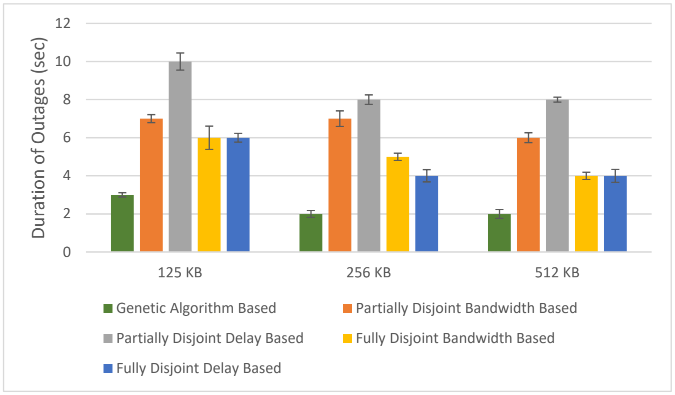
The graphs clearly illustrate that the genetic algorithm-based path selection algorithm outperforms other path selection approaches in the experiments conducted with both the default and DAPS schedulers across different buffer sizes. However, these experiments provide additional insights by demonstrating how different schedulers can impact various QoE parameters differently. Several innovative schedulers proposed in the literature aim to mitigate the HoL blocking issue on heterogeneous paths and improve goodput.

We anticipate that incorporating these various schedulers could enhance the performance of our path selection algorithm. For instance, BLEST reduces the number of retransmissions compared to the default scheduler, thanks to its unique blocking prediction mechanism. Another consideration is the use of ECF. With its notable approach prioritizing the earliest expected completion, ECF facilitates faster data transmission. We expect that the incorporation of BLEST, ECF, or a similar scheduler would contribute to a decrease in the number of quality switches and duration of outages, given their anticipated ability to provide faster transmission and higher goodput. This expectation holds, especially for path approaches that consider the delay differences of the paths. The examination of the comparative test results leads us to the following conclusion: *While recognizing the impact of the scheduler and buffer size on QoE parameters, the correlation between the performances of different path selection algorithms remains unaffected.*

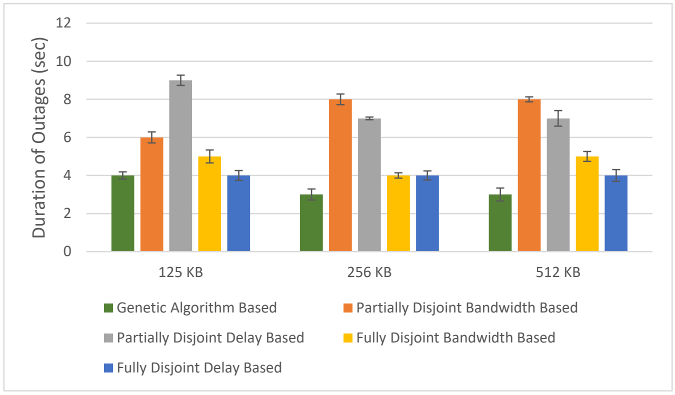
In this study, the whole network setup including the genetic algorithm, SDN network, Mininet, and video clients was implemented on a machine with the Intel® Core™ i7-10750H processor and 16.0 GB Random Access Memory (RAM). The total number of paths varies in the topologies used in the experiments. Therefore, the running time of the genetic algorithm differs for each topology.

In the experiments, we utilized different population sizes, taking into account the scale of the respective network topology. Specifically, the population sizes for the Custom, CompuServe, and Jellyfish topologies were set to 28, 150, and 300, respectively.

We evaluated the impact of different population sizes on



(a) CompuServe Topology, Default Scheduler



(b) CompuServe Topology, DAPS Scheduler

Fig. 9: Average Duration of Outages

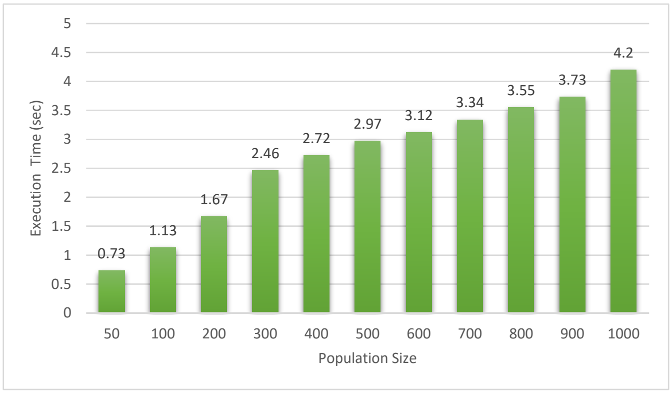


Fig. 10: The run-time performance with various population sizes

execution time by running the genetic algorithm and recording the corresponding running times. Fig. 10 illustrates the execution time of the proposed genetic algorithm across various population sizes. The graph illustrates that the algorithm's running time varied from 0.5 seconds to 2.46 seconds across the different network topologies used in the experiments.

The observed running time of the genetic algorithm proved sufficient for conducting the experiments within our hardware limitations. These findings demonstrate that the proposed approach can yield results even under time constraints imposed by real-time system requirements, particularly when more powerful servers are employed to execute the genetic algorithm.

## V. Conclusion

MPTCP is a promising protocol for increasing the performance of the video streaming application via extended throughput thanks to the use of multiple paths. However, selecting paths for MPTCP subflows is crucial, as the QoE significantly depends on path characteristics.

In this study, we proposed a path selection algorithm for MPTCP subflows by utilizing SDN. The criteria that were considered in the selection of the paths are the bandwidth of the paths, delay differences among the subflows, and the congestion level of the common links used by the subflows. The SDN controller uses a genetic algorithm to select paths.

As mentioned in previous sections, the experiments conducted in this study were performed using the Mininet network emulator on a system with limited processor and RAM capacity. Although Mininet offers a scalable and flexible environment, the capabilities of the host machine influence the performance of the emulator. Therefore, we conducted tests with a maximum of 30 nodes, considering the hardware limitations.

The evaluation of the proposed approach is done by comparing various path selection approaches that consider different criteria. To evaluate, we measure the QoE of the HAS application. The observed QoE parameters show that the delay differences among the MPTCP subflows might be as crucial as the bandwidth values of those paths. We observe that delay differences of the paths have a more negative impact than delay to the outage of durations and average received video bitrate. The main reason is that the video streaming applications that implement adaptive quality changes are highly sensitive to the jitter of the packets that arrive at the client. Different observed latency values lead to fluctuations in throughput calculations, which is critical in the determination of the selected qualities. The wrong decisions about selected qualities might increase the duration of outages.

Although considering the delay differences of the paths provided to enhance the several QoE metrics as the fully disjoint delay-based approach, we observe that the number of quality changes was the highest with that approach. This is another observation that emphasizes the importance of considering bandwidth and the use of partially disjoint paths, to provide improvement in all QoE metrics.

The proposed genetic algorithm-based approach considers two important path selection criteria, bandwidth and delay differences as well as congestion level of common links. It returns the output without affecting negatively any QoE parameters. The experimental results show that the proposed algorithm provides an improvement in all QoE parameters.

It is worth mentioning that there are factors on the client side that are effective on the overall throughput. The network scheduler and the buffer size are two crucial criteria on the client side that we have considered in further evaluations. The preliminary results observed by using two different schedulers and three different buffer sizes indicate that even with good network conditions, there is still room to enhance overall QoE with an improved scheduler. These results also show that, even with the use of a scheduler focusing on diminishing the HoL blocking effect, our proposed path selection algorithm's performance improvement remains consistent compared to other path selection algorithms. However, this work introduces a new research direction, evaluating path selection algorithms alongside various schedulers and considering scheduler design in path selection algorithms. We leave this as a future work.

In future work, we plan to consider the effects of MPTCP schedulers on QoE. This will involve enhancing our genetic algorithm to jointly consider path and scheduler characteristics in path selection at the network layer. Another interesting problem that we plan to work on in the future is the effects of the delay between the switches and the controller for various path selection algorithms. We also plan to enhance our work by considering different types of devices such as TV, desktop, and mobile to select the paths concerning the rendering capabilities of different devices. To evaluate the performance of the proposed approach, we plan to expand our research to encompass various network architectures and conditions, including scenarios with multiple servers, dynamic client patterns, and heterogeneous access networks. Such high dynamism necessitates the rerouting of subflow paths, which presents a distinct and intriguing challenge. While event-based or periodic execution of the genetic algorithm is feasible, determining the optimal frequency requires careful consideration of the communication overhead on both northbound and southbound APIs. To ensure scalability for a large number of clients and a highly dynamic network, we also aim to develop a distributed implementation of the proposed path selection algorithm, enabling it to operate across multiple servers.

## Acknowledgment

The authors would like to express their sincere gratitude to the anonymous reviewers and editor in chief for the constructive review process that has helped improve the quality of the manuscript. The authors would also like to thank Dr. Chi-Dung Phung from Cnam and Dr. Stuart Clayman from University College London (UCL) for their valuable discussions.

## References

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," Tech. Rep., 2013.

[2] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in mptcp," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, 2019.

[3] K. Herguner, R. S. Kalan, C. Cetinkaya, and M. Sayit, "Towards qos-aware routing for dash utilizing mptcp over sdn," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 1–6.

[4] M. Sayit, E. Karayer, C.-D. Phung, S. Secci, and S. Boumerdassi, "Numerical evaluation of mptcp schedulers in terms of throughput and reliability," in *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2019, pp. 1–6.

[5] 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.

[6] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1492–1525, 2018.

[7] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath tcp schedulers," in *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, 2014, pp. 27–32.

[8] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *2013 27th international conference on advanced information networking and applications workshops*. IEEE, 2013, pp. 1119–1124.

[9] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks," in *2016 IFIP networking conference (IFIP networking) and workshops*. IEEE, 2016, pp. 431–439.

[10] 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.

[11] B. Y. L. Kimura, D. C. S. F. Lima, and A. A. F. Loureiro, "Packet scheduling in multipath tcp: Fundamentals, lessons, and opportunities," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1445–1457, 2021.

[12] B.-H. Oh and J. Lee, "Constraint-based proactive scheduling for mptcp in wireless networks," *Computer Networks*, vol. 91, pp. 548–563, 2015.

[13] M. Sandri, A. Silva, L. A. Rocha, and F. L. Verdi, "On the benefits of using multipath tcp and openflow in shared bottlenecks," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 2015, pp. 9–16.

[14] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, and E. Ifeachor, "A novel qoe-centric sdn-based multipath routing approach for multimedia services over 5g networks," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[15] Z. Jiang, Q. Wu, H. Li, and J. Wu, "scmptcp: Sdn cooperated multipath transfer for satellite network with load awareness," *IEEE Access*, vol. 6, pp. 19 823–19 832, 2018.

[16] J. Duan, Z. Wang, and C. Wu, "Responsive multipath tcp in sdn-based datacenters," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 5296–5301.

[17] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "Sdn for mptcp: An enhanced architecture for large data transfers in datacenters," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[18] S. Tariq and M. Bassiouni, "Qamo-sdn: Qos aware multipath tcp for software defined optical networks," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2015, pp. 485–491.

[19] T. Zhang, S. Zhao, and B. Cheng, "Multipath routing and mptcp-based data delivery over manets," *IEEE Access*, vol. 8, pp. 32 652–32 673, 2020.

[20] H. Nam, D. Calin, and H. Schulzrinne, "Towards dynamic mptcp path control using sdn," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 286–294.

[21] M. Morawski and P. Ignaciuk, "A price to pay for increased throughput in mptcp transmission of video streams," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, 2020, pp. 673–678.

[22] N. Kukreja, G. Maier, R. Alvizu, and A. Pattavina, "Sdn based automated testbed for evaluating multipath tcp," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 718–723.

[23] K. D. Joshi and K. Kataoka, "Sfo: Subflow optimizer for mptcp in sdn," in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2016, pp. 173–178.

[24] S. Zannettou, M. Sirivianos, and F. Papadopoulos, "Exploiting path diversity in datacenters using mptcp-aware sdn," in *IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 539–546.

[25] K. Gao, C. Xu, J. Qin, S. Yang, L. Zhong, and G.-M. Muntean, "Qos-driven path selection for mptcp: A scalable sdn-assisted approach," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[26] R. Alvizu, G. Maier, M. Tornatore, and M. Pióro, "Differential delay constrained multipath routing for sdn and optical networks," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 277–284, 2016.

[27] C.-D. Phung, M. Coudron, and S. Secci, "Internet acceleration with lisp traffic engineering and multipath tcp," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2018, pp. 1–8.

[28] O. Abdoun and J. Abouchabaka, “A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem,” *arXiv preprint arXiv:1203.3097*, 2012.

[29] C. H. Benet, A. Kassler, and E. Zola, “Predicting expected tcp throughput using genetic algorithm,” *Computer Networks*, vol. 108, pp. 307–322, 2016.

[30] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, 2021.

[31] H. Li, Y. Wang, R. Sun, S. Guo, and H. Wang, “Delay-based congestion control for multipath tcp in heterogeneous wireless networks,” in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, 2019, pp. 1–6.

[32] P. Cespedes-Sanchez, B. Maluff, D. P. Pinto-Roa, and H. Legal-Ayala, “Hybrid incremental deployment of hsdn devices,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[33] E. Karayer and M. Sayit, “A path selection approach with genetic algorithm for p2p video streaming systems,” *Multimedia Tools and Applications*, vol. 75, no. 23, pp. 16 039–16 057, 2016.

[34] L. H. G. Ferraz, D. M. F. Mattos, and O. C. M. B. Duarte, “A two-phase multipathing scheme based on genetic algorithm for data center networking,” in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 2270–2275.

[35] T. Lu and J. Zhu, “A genetic algorithm for finding a path subject to two constraints,” *Applied Soft Computing*, vol. 13, no. 2, pp. 891–898, 2013.

[36] N. Thamaraikannan and S. Kamalraj, “Utilization of compact genetic algorithm for optimal shortest path selection to improve the throughput in mobile ad-hoc networks,” *Cluster Computing*, vol. 22, no. 2, pp. 3715–3726, 2019.

[37] 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.

[38] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “Tcp extensions for multipath operation with multiple addresses,” Tech. Rep., 2013.

[39] S. Kamath, A. Srivastava, P. Kamath, S. Singh, and M. S. Kumar, “Application aware multiple constraint optimal paths for transport network using sdn,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4376–4390, 2021.

[40] M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei, “The generalized bin packing problem,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 6, pp. 1205–1220, 2012.

[41] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX, 2010.

[42] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, “Jellyfish: Networking data centers randomly,” in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012, pp. 225–238.

**Shadi Bikas** received her B.S. in Computer Science from Orumiyeh Payamnour University, and M.S. degrees in information technology in 2013 and 2019 respectively. Moreover, she is a Ph.D. candidate in Information Technology at Ege University. Her research interests mainly include Software Defined Networking and Multipath TCP as well as Reinforcement Learning and Deep Learning.

**Müge Sayıt** received her M.Sc. degree in 2005 and a Ph.D. degree in 2011 in Information Technologies from the International Computer Institute at Ege University in Turkiye. After working as an assistant professor at the same institute, she started serving as an associate professor in 2017. Currently, she works at the School of Computer Science and Electronic Engineering at the University of Essex, UK. Her research interests include Software Defined Networking, Network Function Virtualization, future networks, video streaming, and video codecs. She has been working as the principal investigator or as a researcher in various R&D projects.