QoS-aware Multipath Routing in Software-Defined Networks

Priyanka Kamboj, Sujata Pal, *Senior Member, IEEE*, Samaresh Bera, *Member, IEEE*, and Sudip Misra, *Fellow, IEEE*

***Abstract*—The emergence of new applications, such as online gaming and virtual reality, necessitates the underlying network capable of fulfilling high bandwidth and low latency requirements. The software-defined multipath routing is a viable approach to fulfill such quality-of-service (QoS) requirements by improving the data delivery performance through multipath. In this paper, we propose a QoS-aware dynamic multipath routing scheme for enhancing QoS of high-bandwidth applications in an SDN-enabled network. The proposed scheme consists of three phases – flow splitting, multipath routing, and flow reordering. In the first phase, we propose a flow splitting scheme to decide how to split the incoming flows to enable multipath routing in the network. In the second phase, we design a cost function for routing the splittable subflows and formulate a min-cost routing problem as an integer linear program (ILP). To solve the problem in polynomial time, we propose a greedy heuristic approach. Finally, in the third phase, we propose a flow reordering scheme for the received subflows through multiple paths to maintain the desired flow sequence at the destination. The experimental results show that the proposed scheme achieves higher network throughput by 22% compared to the benchmark schemes. Further, the proposed scheme achieves a reduction in QoS violated flows by 24% compared to the benchmark schemes.**

***Index Terms*—Multipath routing, Optimization, Quality-of-service, Software-defined networking**

I. Introduction

THE significant advancement of emerging applications, such as online gaming, virtual reality, and vehicle-to-everything (V2X), increased data traffic in the network. The quality-of-service (QoS) requested by these applications ranges from high bandwidth to low latency and high reliability. Therefore, it becomes necessary to achieve a high packet transmission rate with low packet drop and low latency for end-user applications [1], [2]. However, the present Internet architecture is incapable of meeting such diverse and stringent QoS requirements by these applications [3]–[5]. Software-defined networking (SDN) is a novel technology that addresses the limitations of the current Internet architecture through simplified and flexible network management [6], [7].

SDN provides network abstraction by separating the control functions from the network devices. Thus, it provides a global view of the network and yields an improved network utilization [8], [9]. Further, the dynamic configuration of QoS policies enables the SDN controller to make forwarding decisions based on application-specific requirement [10], [11].

In this work, we focus on multipath routing in SDN-enabled network to meet the high bandwidth requirements of the emerging applications, as mentioned above. There exists a few works that focus on multipath routing in SDN [11]–[13]. Recent studies [12]–[19] explored the benefits of multipath transport protocols and enhanced the performance of network applications. While the existing works are useful for multipath routing, they have the following limitations. First, the packet scheduler distributes the data packets without head-of-line blocking delays with transport layer multipath routing [15]. Second, most works perform an uneven packet distribution over multiple paths. As a result, the data packets can reach out-of-order at the destination due to the uneven distribution [20], [21]. However, the existing works do not consider these issues while routing the data packets through multipath. Consequently, it becomes necessary to address the following questions for multipath routing of data packets – a) how to split the flows through multiple paths, b) how to forward the splittable flows, and c) how to reorder the out-of-order packets received at the destination.

Fig. 1 illustrates the issues with multipath routing. Fig. 1a presents that a single path routing fails to meet the bandwidth requirements. Fig. 1b shows that the traffic flows are split and routed through different paths using a routing algorithm to meet the bandwidth requirements. The packets reach with out-of-order sequences at the destination; therefore, subflows need to be reordered at the destination node, as shown in Fig. 1c.

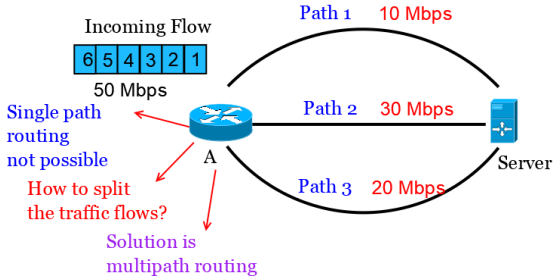
In this paper, to address the issues mentioned above, we propose a QoS-aware multipath routing scheme in SDN-enabled network. The proposed scheme consists of three phases – splitting incoming flows, routing path selection, and reordering received flows. In the first phase, we propose a flow-splitting algorithm to split the incoming flows among multiple paths. In the second phase, we formulate a min-cost integer linear program (ILP) to route the splittable flows through multiple paths. To solve the ILP in polynomial time, we propose a greedy-heuristic approach. The split flows over multiple paths may be received out-of-order at the destination. Consequently, in the third phase, we propose a flow reordering scheme. In brief, the significant contributions in this work are as follows:

- We propose a scheduling scheme to enable multipath routing to improve QoS in SDN-enabled networks. The proposed scheme splits and schedules the traffic flows

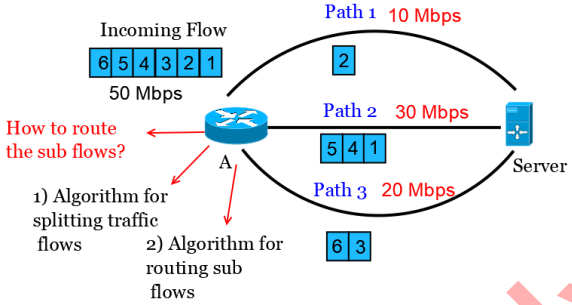
P. Kamboj and S. Pal are with the Department of Computer Science and Engineering, Indian Institute of Technology, Ropar, India, 140001. (E-mail: prinkskamboj12@gmail.com, sujata@iitrpr.ac.in).

S. Bera was with the Dept. of CSE, IIT Kharagpur, 721302, India when the work was done. He is currently with the Department of ECE, IISc Bangalore, India, 560012. (Email: s.bera.1989@ieee.org).

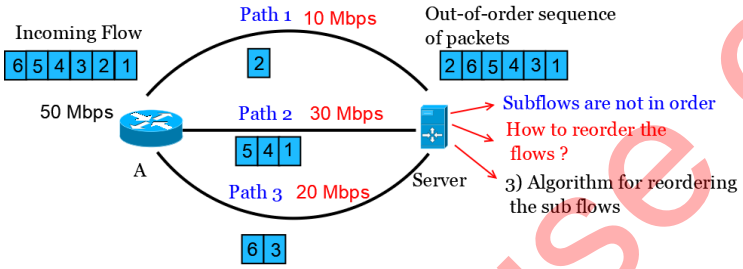
S. Misra is with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India, 721302. (E-mail: smisra@cse.iitkgp.ac.in).



(a) A request with a data-rate requirements of 50 Mbps arrives at switch A which needs to be routed to the server. The single path routing fails as it cannot meet the required data-rate.



(b) Request has been allocated a data-rate of 50 Mbps using multipath routing



(c) Subflows through multiple paths arrived out-of-sequence; packet reordering needs to be done.

Fig. 1: Illustrative example: a) single path routing fails to meet the data-rate requirements; b) flows are split and routed through different paths to meet data-rate requirements; and c) the split flows received at the destination need reordering.

among multiple end-to-end disjoint paths while considering throughput requirements by the applications.

- We design a cost function for routing the split flows. We formulate a min-cost routing problem as an integer linear program (ILP) while considering associated constraints. As the ILP turns out to be NP-hard, we propose a greedy-heuristic approach to find the solution for the QoS routing problem in SDN.
- We design a flow reordering scheme for the received subflows through multiple paths to deal with out-of-order packets at the destination.
- We evaluate the performance of the proposed scheme using the Mininet network emulator and the Ryu SDN controller. The proposed approach achieves higher network throughput, lower delay, and lower QoS violated flows than the state-of-the-art schemes.

We organize the rest of the paper as follows. In Section II, we discuss the existing approaches to QoS-aware single path routing in SDN and multipath routing. Section III presents the prerequisites and system architecture. Section IV describes the proposed QoS-aware multipath routing approach. Section V discusses the experimental results. In Section VI, we highlight the practical applications of the proposed scheme. Finally, in Section VII, we conclude the paper while highlighting the future research directions.

II. Related Work

The researchers have proposed different schemes to enhance QoS in SDN. In this section, we classify the related work from three aspects – QoS management in SDN [22]–[25], resource reservation using multipath approach [2], [26]–[28], and traffic scheduling using multipath routing approach [1], [13], [15]–[17], [29], [30]. In the following subsections, we discuss the existing approaches.

A. QoS Management in SDN

Recent studies [22]–[25] had highlighted the traffic routing schemes in SDN in terms of resource reservation and enhancing QoS. Egilmez *et al.* [22] proposed a QoS guaranteed data delivery in multimedia applications. They proposed a prioritization scheme to route the multimedia traffic flows in SDN networks. The authors employed per-flow routing, and the scope of their proposed method was limited to single-path routing. Sharma *et al.* [23] introduced a QoS framework for inter-domain routing where each domain was regulated using an SDN controller to prioritize the business traffic over best-effort traffic. The framework focused on assigning bandwidth to the incoming request using a bandwidth broker. But, they did not schedule the traffic flows onto multiple paths. Saha *et al.* [24] presented an SDN-based routing solution to route the heterogeneous traffic categorized as delay- and loss-sensitive flows to meet application-specific QoS requirements. Further, Kumar *et al.* [25] introduced a framework to route high critical flows having end-to-end delays and bandwidth requirements in real-time systems. The framework isolates flows into different queues for each traffic class to ensure QoS guaranteed traffic forwarding.

B. Resource Reservation using Multipath Routing

In comparison with single-path routing, multipath routing improves the network throughput. Yan *et al.* [26] introduced the method that generates multiple paths and uses queuing mechanisms to provide bandwidth guarantee and enhances QoS to different traffic classes. Sahhaf *et al.* [27] proposed an adaptive multipath provisioning method that uses a time-slot-based approach to select multiple paths having maximum bandwidth and availability. Huang *et al.* [2] introduced a utility-based model in hybrid SDN for flow-level bandwidth allocation over multiple paths. The authors in [28] had proposed a new approach to route delay-sensitive traffic flows in the network. Their approach performs network monitoring to compute parameters such as delay and throughput to measure QoS and presents a probabilistic-matching technique for traffic distribution over multiple paths. Bagaa *et al.* [30] reduced the operational expenditure costs and thus proposed a QoS-aware multipath routing in SDN to allocate network resources.

C. Traffic Scheduling with Multipath Routing

To deal with bandwidth bottleneck in traffic forwarding on a single path, the authors in [16] proposed a multipath routing algorithm for load balancing in data centers. The controller schedules the flows with dynamic link costs on the least loaded paths. Jiawei *et al.* [1] introduced a priority-based multipath routing algorithm for SDN networks. The algorithm allocates network resources to maximize their utilization while considering the flow priority of multimedia traffic. Tuncer *et al.* [31] implemented a traffic split mechanism using the IP addresses and uses *Per-Flow* multipath routing method.

Farrugia *et al.* [29] proposed a per-packet flow splitting mechanism to handle unequal split ratios while considering scalability in the SDN network. The routing table in the SDN switch includes the series of output port numbers with their corresponding split ratios. The authors in [13] present a multipath routing scheme to meet bandwidth requirements of multimedia applications using segment routing in SDN networks. The server assigns multiple disjoint routing paths to real-time service flows by minimizing packet disordering and reducing transmission latency to improve end-user quality of experience (QoE). Wang *et al.* [32] proposed a network virtualization scheme for multipath routing using SDN. The scheme provides resources for flow splitting and performs tag-based forwarding for packet reordering at the destination node.

Synthesis: Existing approaches do not provide the aggregated capacity to meet the bandwidth requirements of traffic flows. The detailed synthesis of the existing work suggests that a research gap exists to minimize the routing cost in the network. Further, the packet reordering at the destination is not considered while routing the subflows. Therefore, scheduling traffic flow onto multiple paths to enhance QoS for end-user applications requires more attention. This work considers end-to-end traffic scheduling in the SDN-enabled network to propose a packet scheduling method, path selection, and packet reordering in the network. We consider a single SDN controller to control the end-to-end network in this work. However, we can also use multiple controllers in a coordinated manner to achieve this. We limit the discussion on using multiple SDN controllers in this work as the primary objective is traffic routing in the network.

TABLE I: List of notations

Notation	Description
$\mathcal{V}$	Set of switches
$\mathcal{L}$	Set of links
$\mathcal{F}$	Set of flows
$F_{u,v}$	Set of flows over a link $(u,v) \in \mathcal{L}$
f	A flow in the network
F_{spl}	Set of splittable flows
s_f, t_f	Source and destination node
P_f	Set of paths for flow f
b_f	Bandwidth requirement by a flow f
$c_{u,v}$	Bandwidth capacity of a link $(u,v) \in \mathcal{L}$
$c_{u,v}^{util}$	Link capacity utilization of a link (u,v)
$d_{u,v}$	Delay of a link $(u,v) \in \mathcal{L}$
$d_{u,v}^{max}$	Maximum tolerable delay on a link $(u,v) \in \mathcal{L}$
d_f^{max}	Maximum delay tolerable by a flow $f \in \mathcal{F}$
π_f^p	A binary variable represents whether path p is chosen for a particular flow f
$\Phi_{u,v}^f$	Cost of routing a flow $f \in \mathcal{F}$ over a link $(u,v) \in \mathcal{L}$

III. Prerequisites

We present the network in the form of a directed graph $G(\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ denotes the set of SDN switches, and $\mathcal{L}$ indicates the set of links between the switches. Each link has a bandwidth capacity $c_{u,v}$ and delay $d_{u,v}$ between nodes u and v. The user requests are forwarded through the network. Let there be a set of flows associated with different applications denoted as $\mathcal{F} = \{1, 2, ..., N\}$, where each flow has a bandwidth requirement and delay-bound. Table I represents the key notations that are used in the paper. The set of paths P_f for a flow $f \in \mathcal{F}$ is described as:

$$P_f = \left\{p_f^1, p_f^2, ..., p_f^n\right\}, \quad \forall f \in \mathcal{F}. \tag{1}$$

In this work, we consider that all flows are splittable. A flow is called splittable if it can be split into multiple subflows and each subflow can be routed in the network from the source to the destination independently.

Thus, the paths needed for a distinct flow has to satisfy the following constraints:

$$\sum_{p \in P_f} \pi_f^p \geq 1, \quad \forall f \in F_{spl}, \tag{2a}$$

where (2a) represents that the splittable flows can have more than one path. Further, π_f^p is the binary variable that represents whether path p is chosen for a particular flow f.

Fig. 2 represents the SDN controller modules, where highlighted modules in blue color are proposed in this work. The *Packet-in Handler* module captures the packet-in messages from the switches. The *flow split manager* module is implemented at the source switch. This module takes the input as packet-in messages from the *Packet-in Handler*. Several methods have been proposed in the literature for traffic splitting, such as packet-based, flow-level, and sub-flows-based [15]. In the literature, the flow-based approach uses hashing techniques, whereas packet-level is based on the Round Robin (RR) approach [15], [28]. This work proposes a new flow

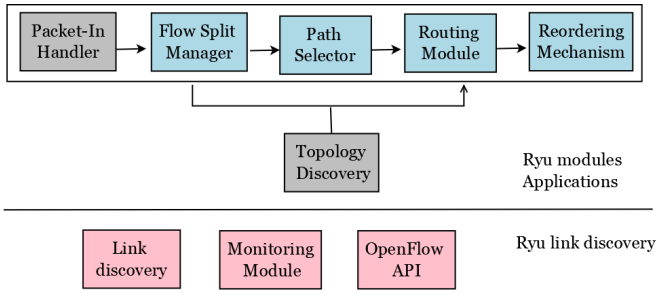


Fig. 2: Proposed system architecture

splitting mechanism that divides the traffic flow into numerous flow units and is forwarded through different paths. We assume that the flow split mechanism is executed at the sender side, where traffic scheduling for paths is determined using the packet scheduler.

The ingress (or incoming) switch performs the task of flow splitting, whereas traffic aggregation takes place at the egress (or outgoing) switch. The ingress switch receives the packet and assigns a packet sequence number (PSN), flow ID, and priority number. The PSN maintains the packet ordering, whereas flow ID helps differentiate the traffic flow type. The packet is routed on different paths and differentiated using flow ID. The priority defines the packet forwarding order in the network. The controller gives the flow request with a higher priority preference than the lower priority traffic flows. Finally, the egress switch removes the PSN, flow ID, and priority tag from the packet upon its reception and forwards it to the destination node.

IV. Proposed Multipath Routing in SDN

Objective: Given the network with link capacity, delay, and a set of flows with their bandwidth requirements and delay-bounds. The objective is to route flows as many as possible from source to destination through multiple paths. We pose this problem by addressing the following questions – how to split the flows among multiple paths, how to route the split flows, and how to reorder the out-of-order flows at the destination. We address these issues in the following sections.

A. Flow Splitting

The task of the *scheduling algorithm* is to assign each flow unit to a different port based on the scheduling policy. The *scheduling algorithm* adapts based on the varying traffic flows in the network. We utilize the OpenFlow protocol to implement flow-splitting using group tables that help to forward decisions on several links [29].

We aim to maximize the performance of the network to deliver end-user applications. The scheduler adapts to the change in path characteristics based on the bandwidth and delay of the link to utilize available capacity efficiently and avoid overloading the congested path. To maximize the throughput of end-user applications, we adopt a Weighted Round-Robin scheduler (WRR) that distributes a stream of packets along different paths [15]. The weight of traffic associated with each path has been computed dynamically by the scheduling algorithm as follows:

$$w_{u,v} = \frac{c_{u,v}}{\sum_{(u,v)\in neigh(i)} c_{u,v}}, \forall i \in \mathcal{V}, \tag{3}$$

where $w_{u,v}$ represents the weight of the link (u, v), and $c_{u,v}$ denotes the available bandwidth of the link (u, v). Fig. 3 shows an illustrative example of flow-splitting. The value on the links represents the available bandwidths. The flow at switch 1 splits according to the weight function presented in (3). Algorithm 1 represents the flow splitting and probabilistic matching. For probabilistic matching, the weight of each bucket is limited between 1 to 100. Further, the sum of their weights is equivalent to 100.

Consider an example with three paths in the network topology to reach the destination node. The switch wants to send 30% of the traffic packets on port 1, 15% over port 2, and the rest 55% through port 3. We consider that are three buckets with weights of 0.30, 0.15, and 0.55, as shown in Figure 3. When a switch receives a flow, a random number is generated, for instance, 0.40. Further, mapping is done with the bucket weight as shown in Algorithm 1. The corresponding bucket action is chosen to send packets to that port.

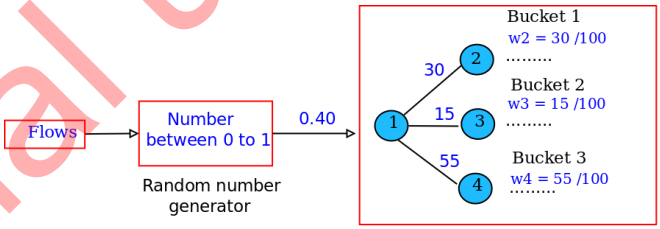


Fig. 3: Illustration example to compute bucket weight

Algorithm 1 Flow Splitting and Probabilistic matching

1: Compute weights for each bucket and sort their weights (using (3))
2: **while** matching of packet takes place **do**
3: Generates a random number n whose value lies between 0 to 1
4: Action is taken on performing a match of n with each bucket weight
5: **end while**

B. Routing Path Selection

In Section IV-A, a flow is split into sub-flows, and each subflow needs to be routed in the network. In this work, we focus on minimizing the routing cost associated with flows while considering the associated constraints. Therefore, the objective is to maximize the number of flows to be forwarded through the network while considering the constraints. Consequently, we design the cost function consisting of delay and bandwidth requirements of flows and available networking resources. The cost function $\Phi_{u,v}$ determines the cost of link (u, v) as follows:

$$\Phi_{u,v} = \alpha \frac{c_{u,v}^{util}}{c_{u,v}} + (1-\alpha)\frac{d_{u,v}}{d_{u,v}^{max}}. \tag{4}$$

where $d_{u,v}$, $c_{u,v}$ represents delay and available bandwidth of a link (u, v). Further, $d^{max}_{u,v}$ denotes the maximum delay tolerable and $c^{util}_{u,v}$ represents link capacity utilization of a link (u, v). The parameter α is designed to consider the impact of link capacity and delay on the cost function.

Optimization problem: The centralized controller knows the entire network state and computes the forwarding table to route the data from one switch (node) to another switch. The controller updates the forwarding table and stores it on the network switches in case of any failure. Once the flows are split across multiple paths, the controller determines the forwarding path for each subflows. Mathematically,

$$\min \sum_{f \in \mathcal{F}} \sum_{(u,v) \in \mathcal{L}} f_{u,v}\ \Phi_{u,v}, \quad \forall (u,v) \in \mathcal{L} \tag{5a}$$

$$\textbf{s.t.} \sum_{u:(u,v)\in\mathcal{L}} f_{u,v} \; - \sum_{u:(v,u)\in\mathcal{L}} f_{v,u} = \begin{cases} 1 & u = s_f, \\ 0 & \forall u \neq s_f, t_f, \\ -1 & u = t_f \end{cases} \tag{5b}$$

$$f_{u,v}\ b_f \leqslant c_{u,v}, \quad \forall (u,v) \in \mathcal{L} \tag{5c}$$

$$\sum_{(u,v)\in P_f} d_f(u,v)\ f_{u,v} \leq d^{max}_f(u,v), \quad \forall (u,v) \in \mathcal{L} \tag{5d}$$

$$f_{u,v} \in \{0, 1\}\,. \tag{5e}$$

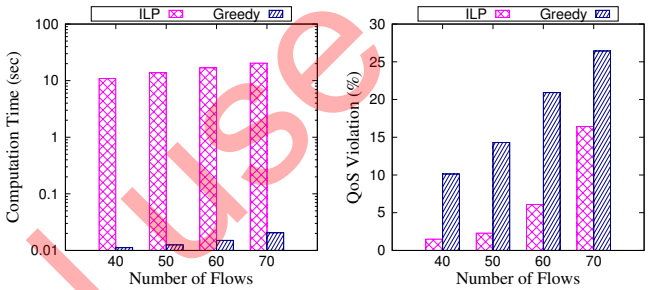
Equation (5b) states the flow conservation constraints. Equation (5c) represents that the traffic flows on a link should not exceed the link capacity. Equation (5d) denotes the *delay constraints*, where d^{max}_f represents the delay-bound of a flow $f \in \mathcal{F}$. Equation (5e) states the binary variable on whether link $(u, v) \in \mathcal{L}$ is selected to route flow f. The optimization problem is NP-hard. We limit the discussion on the NP-hard problem as it is well-examined in the existing literature [17]. We design a greedy heuristic-based solution to compute the optimal routing paths as follows:

Algorithm 2 is the *Multiple Routing Path (MRP)* algorithm that augments the traffic flows on the path computed using the *Bandwidth_allocation* algorithm presented in Algorithm 3. Algorithm 3 calculates the shortest routing paths to the user by satisfying the QoS demand while ensuring end-to-end QoS. The *MRP* algorithm considers a set of flow requests represented by $\mathcal{F}$. The *MRP* algorithm calls the *Bandwidth_allocation* algorithm to process each user request in Step 4.

In the MRP algorithm, we use the K-shortest path algorithm [33] to compute the disjoint routes in the network topology. We sort the paths based on the cost function calculated in (4). We select the least cost routing path for traffic flows in the network using (5a). Since the proposed approach assigns a lower cost to the links with higher bandwidth, which facilitates the selection of the least loaded links, thus, we can say that congestion control is inbuilt into the network.

The *VERIFY_QOS* function in the *Bandwidth_allocation* algorithm checks the bandwidth requirements of the user's request and prefers the best-path to route traffic flows. The *Bandwidth_allocation* algorithm returns the "best-path" to the *MRP* algorithm. The *MRP* algorithm augments the traffic flows on the best-path. Further, after augmenting traffic flows, the residual link capacity of the path and residual graph are updated, as presented in Steps 7-16 in Algorithm 2.

We used cvxpy [34] to solve the ILP. Fig. 4 shows a comparison between the proposed greedy approach and the ILP solution in a scale-free Barabasi-Albert network topology [35]. In the network, we consider 10 nodes with average degree as 2. The details of the delay and bandwidth parameters of the network and flows are mentioned in the simulation settings in Section V. It is evident that the proposed greedy approach yields competitive performance to the optimal solution in terms of QoS violation, while requiring very less computation time.



(a) Computation Time (b) QoS Violation

Fig. 4: Comparison between ILP and proposed scheme

Algorithm 2 Multiple Routing Paths (MRP)

Input : Graph G, b_f ▷ User-defined ▷ b_f represents bandwidth demand of user request

Output : Flow_request is fulfilled or not

1: $k = 1$, flag = 0
2: $\mathcal{F}$ = Set of flows
3: **while** $f_k \in \mathcal{F}$ not been forwarded **do**
4: arr(flag, best-path) = *Bandwidth_Allocation* (*flow_request*, b_f, G); ▷ Using Algorithm 3
5: **if** (flag == 1) & best-path **then**
6: Augment the traffic flow on path
7: **else**
8: No QoS path not found
9: **end if**
10: *UPDATE_LINK_CAPACITY (best-path, b_f)*
11: $k \leftarrow k + 1$
12: **end while**
13: **function** UPDATE_LINK_CAPACITY(P, b_f)
14: **for** (u, v) in P **do**
15: $c_{u,v}$ = $c_{u,v}$ - b_f ▷ Capacity is reduced by bandwidth demand of flow
16: **end for**
17: **return** request_status;
18: **end function**

Algorithm 3 Bandwidth_Allocation

Input : Graph G, source s_f, destination d_f, $c_{u,v}$, b_f ▷ User-defined ▷ b^f represents bandwidth demand of user request

Output : Path P on which flow_requests $\mathcal{F}$ are routed

for P in K-shortest-path (s_f, t_f) **do**
  Sort the paths using cost function computed in (4)
  if *VERIFY_QOS* (P, b_f) **then**
    flag = 1
    best-path = P
    return (flag, best-path);
  end if
end for
function VERIFY_QOS(P, b_f)
  for (u, v) in P **do** ▷ QoS satisfied
    if $c_{u,v} \geq b_f$ **then**
      return True
    else
      return False
    end if
  end for
end function

Algorithm 4 Flow Detection and Recovery

Input : Set of flows $\mathcal{F}$, *PSN*, *Flow Id*

Output : Flows are recovered based on *Flow Id* and *PSN*

while destination node receives flows **do**
  send them to buffer corresponding to flow
end while
while there is out-of-order packets **do**
  Compute m_d = RI - AS;
  if $|m_d|$ = 0 **then**
    No packet reordering;
  else
    if $|m_d| > 0$ && $|m_d| < 2$ **then**
      FLOW_RECOVERY(Flow Id, PSN);
    else
      Packet is considered to be lost;
    end if
  end if
end while
function FLOW_RECOVERY(*Flow Id, PSN*)
  while flow recovery checks PSN of flows **do**
    Perform packet reordering by sorting flows by PSN and Flow Id;
  end while
end function

C. Reordering out-of-order Flows

The issue of packet ordering at the destination arises from the traffic routing on multiple paths. Thus, we design and implement the flow recovery and packet reordering mechanism at the destination node to reduce its impact on the performance. Let the packet sequence number (PSN) to the packets is assigned as $\{1, 2, ..., N\}$. We introduce a *resequencing buffer* at the destination node for each flow. We assumed the arrived sequence (AS) number to the incoming packets as $\{1, 2, ..., M\}$. Further, the destination node assigns a received index (RI) to each packet upon its reception as $\{1, 2, ..., S\}$.

Algorithm 4 checks the early or late arrival of the packet. We compute a mean displacement (m_d) at the destination node as the difference between the AS and RI. If m_d = 0 then no ordering of packets are required. The value of $m_d > 0$ and $m_d < 2$ indicates the late arrival and early arrival of the packet respectively. We consider a threshold value as 2 ($m_d > 2$) for the lost packets in the network. We assume that if a packet AS does not receive up to the third consecutive sequence number, thus packet is considered lost. The algorithm invokes a *FLOW_RECOVERY* function to perform packet reordering. This function checks the PSN of the flows and sorts the flows at the destination node based on the PSN and the flow ID, as explained in the flow split manager.

D. Running Time Complexity

We analyze the worst-case time complexity of the proposed approach – flow splitting and probabilistic matching (Algorithm 1), routing path selection (Algorithm 2, Algorithm 3), and flow detection and recovery (Algorithm 4). Algorithm 1 involves sorting the bucket weights. Therefore, it takes the execution time as $O(|b|)$, where b is the number of buckets. In Algorithm 2, for each flow request $f \in \mathcal{F}$, the while loop makes a call to the Algorithm 3 in Step 6. Algorithm 2 for each incoming request makes a call to *Bandwidth_Allocation* function. In Algorithm 3, the shortest paths computed using the K-shortest path algorithm are the most expensive operation. The running time of computing K-shortest paths is $O\left(k\left|\mathcal{V}\right|\left(\left|\mathcal{L}\right|+\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)\right)$. Since Algorithm 2 invokes the Algorithm 3 for $|\mathcal{F}|$ number of flow requests, the Algorithm 2 runs in $O\left(\left|\mathcal{F}\right|\left(k\left|\mathcal{V}\right|\left(\left|\mathcal{L}\right|+\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)\right)\right)$. Finally, in Algorithm 4, the while loop performs sorting of flows by Flow Id. Therefore, it runs in $O(|\mathcal{F}|^2 + |\mathcal{F}|)$ time, where $|\mathcal{F}|$ is the number of flows. So, the total time complexity of the proposed scheme is $O\left(\left|\mathcal{F}\right|\left(k\left|\mathcal{V}\right|\left(\left|\mathcal{L}\right|+\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)\right)\right)$ + $O\left(k\left|\mathcal{V}\right|\left(\left|\mathcal{L}\right|+\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)\right)$ + $O(|\mathcal{F}|^2 + |\mathcal{F}|) \approx O\left(\left|\mathcal{F}^2\right|\left(k\left|\mathcal{V}\right|\left(\left|\mathcal{L}\right|+\left|\mathcal{V}\right|\log\left|\mathcal{V}\right|\right)\right)\right)$.

V. Performance Evaluation

A. Simulation Settings

We evaluate the performance of the proposed method using Mininet [36] network emulator and Ryu[1] SDN controller. A Mininet emulator is used for modeling a virtual network to perform the simulation environment. The work experiments have been performed using the HP-ProDesk i7 CPU, 3.62 GHz processors, and 8 GB RAM. The OpenFlow 1.3.1 protocol is used in OpenFlow switches to route the traffic flows in the network. The OpenFlow switches are used to set up multiple paths, and bandwidth is controlled with the help of their meter function. We used the Iperf tool[2] and Distributed Internet Traffic generator (D-ITG) [37] to model the flow requests in

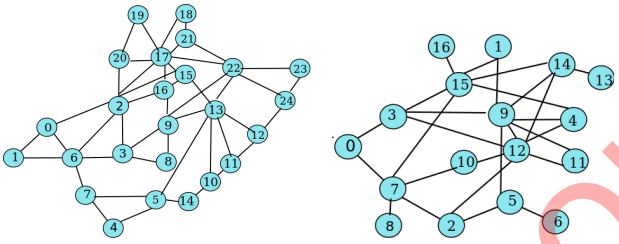
[1]https://ryu-sdn.org/
[2]http://software.es.net/iperf/

TABLE II: Simulation Settings

Parameter	Value
Topology	AttMpls, Goodnet
Number of switches	25 (AttMpls), 17 (Goodnet)
Number of links	57 (AttMpls), 31 (Goodnet)
Number of flows	50 - 250
Flow bandwidth	0.30 - 0.60 Kbps
Average packet size	94 - 699 bytes [38]
Mean rate	562 – 516,540 bps [38]
Active time	1 - 50 s

the network. We adopted the method of uniform distribution to generate packets for the experiment.

To conduct the simulation, we examine two network topologies – AttMpls and Goodnet from the Internet topology zoo [35]. The AttMpls topology and Goodnet topology consist of 25 switches and 17 switches, respectively, as depicted in Fig. 5. Circles represent the switches in the network topology, and they are named 0, 1, 2, and 3. AttMpls topology is large and denser than Goodnet topology. Moreover, the number of nodes, links, and the incoming degree of topology depicts the reason for choosing these topologies to be a favorable candidate for the experimental analysis of the proposed scheme. Table II represents the simulation parameters used in the experimental work.



(a) AttMpls Topology (b) Goodnet Topology

Fig. 5: Network topologies used in experiment

B. Benchmark Schemes

We compare the proposed approach with the following benchmark schemes: Multi-path SDN (MPSDN) proposed in [15], priority-based dynamic multi-path routing (PDMR) proposed in [1], and Multi Constraint Optimal Path (MCOP) architecture presented in [39]. The MPSDN scheme proposed the architecture to schedule and reorder traffic on multiple heterogeneous paths. The scheme uses Dijkstra's algorithm with a minimum priority queue to discover the routing path for traffic flows. The PDMR scheme allocates network resources to a set of flows according to traffic flow requirements and finds the routing path using the cost function for different traffic classes. The MCOP scheme selects the multiple optimal paths to forward the packets while considering the QoS requirements of traffic flows.

On the contrary, the proposed approach evaluates application performance and routes traffic requests (or flows) on routing paths computed using the K-Shortest Path algorithm. The algorithm schedules the traffic flow into multiple paths. It also proposes a flow recovery mechanism to solve the packet reordering problem due to the difference in delay in multiple paths.

C. Performance Metrics

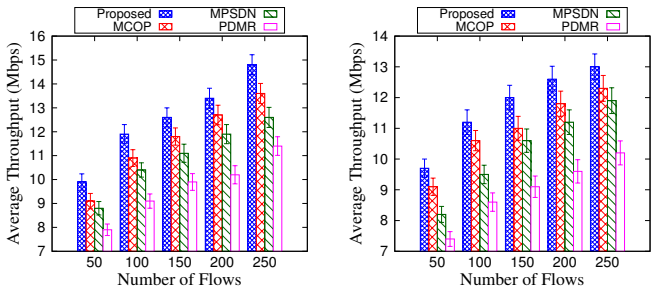
We examine the below metrics in work to estimate the performance of the proposed approach.

- *Throughput:-* It is computed by the bandwidth utilization of links that are active while forwarding the traffic flows.
- *QoS Violated Flows:-* A flow is considered a QoS violated flow if it does not fulfill atleast one of the requirements – link capacity, end-to-end delay, and packet loss while forwarding the traffic flows in the network. In the proposed scheme, QoS violated flows depend on the link capacity utilization and path selection while routing the subflows.
- *Packet Loss:-* The packet loss (or drop) in the network is calculated by the proportion of packets lost to the total packets sent in the network topology.
- *End-to-End Delay:-* We computed the average end-to-end delay experienced by traffic flows in the network.

The throughput, packet loss, and end-to-end delay are measured by the utilities available at the Mininet. Whereas we measure the QoS violation at the Ryu controller considering application-specific requirements.

D. Results and Discussion

This section presents the simulation results obtained in two network topologies – AttMpls and Goodnet with different flow requests. To evaluate the performance of the obtained results, we compare the proposed approach with the mentioned benchmark schemes in terms of network throughput, end-to-end delay, rejection rate, and packet loss. The average of different metrics of the simulation results has been taken for an iteration of 20 runs to compute the value of each parameter. Additionally, we computed the variance of the results at the 95% confidence intervals.



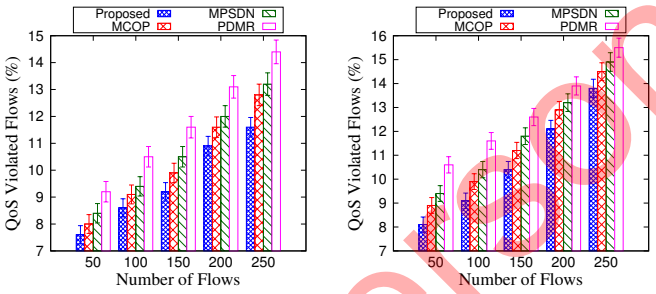
(a) AttMpls Topology (b) Goodnet Topology

Fig. 6: Average throughput in the network with increasing number of flows

1) Throughput: Fig. 6 shows the improvement in average throughout as compared to the existing schemes. The average throughput of all the schemes increases with the rise in the number of flows. Notably, the proposed multipath routing approach demonstrates better performance than benchmark methods in average throughput with the surge in flows. In particular, the proposed scheme achieves a higher network

throughput by 7%, 18%, 27% (with AttMpls topology) and 6%, 16%, 25% (with Goodnet topology) as compared to MCOP, MPSDN, and PDMR schemes, respectively.

In particular, in Fig. 6, we observe that the PDMR scheme attains the lowest throughput as compared to other schemes in both AttMpls and Goodnet topology. The reason for this is that this method does not consider link utilization. PDMR allocates network resources using the link cost function and uses a technique to aggregate bandwidth resources to optimize link weights. Further, we observe that AttMpls topology attains higher average throughput than Goodnet topology because it is larger and denser than Goodnet. Therefore more paths are available to meet the bandwidth demand of user requests. However, the MPSDN scheme attains higher throughput than PDMR. This scheme uses an estimated available bandwidth between the nodes to maximize the overall throughput. The MCOP scheme achieves higher throughput than MPSDN and PDMR schemes. The scheme uses most of the available bandwidth on the link, resulting in increased throughput. On the other hand, the proposed multipath routing approach performs better than benchmark methods in average throughput in both network topologies. The proposed approach considers link capacity and forwards the traffic flows on available maximum residual capacity paths. The scheduler schedules the traffic flows on multiple paths to meet the bandwidth demand of the user requests. Thus, the proposed scheme satisfies the QoS requirements and is scalable to large flows.

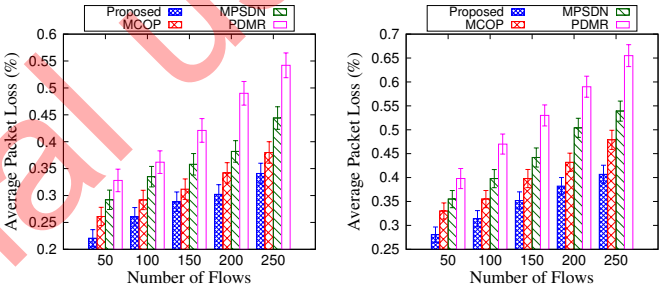


(a) AttMpls Topology (b) Goodnet Topology

Fig. 7: Impact of the QoS violated flows in the network with increasing number of flows

2) QoS Violated Flows: Fig. 7 depicts the performance of the proposed multipath routing scheme that has been achieved in comparison to the existing schemes. We notice that with the rise in the flows in the network, the proposed approach shows better performance than benchmark methods in QoS violated flows. In particular, with 250 flows, the proposed scheme reduces QoS violated flows by 7%, 12%, 22% (with AttMpls topology) and 9%, 14%, 26% (with Goodnet topology) as compared to MCOP, MPSDN, and PDMR benchmark methods respectively.

From Fig. 7, we notice that the PDMR scheme has the lowest performance and has a higher QoS violated flows with the increase in the number of flows in both AttMpls and Goodnet topologies. The reason is that the PDMR scheme allocates network resources using the link cost function for different user requests and does not consider link utilization. Due to this, the resources of the links are exhausted rapidly, leading to more QoS violated flows. However, the PDMR scheme's capacity to perform load balancing is limited because the link cost function only considers metrics jitter and packet loss on each path. Moreover, in the Goodnet topology, the QoS violated flows are higher than in the AttMpls topology. As the Goodnet topology is sparser than AttMpls, thus it has fewer alternative paths to route the traffic flows. On the contrary, the MPSDN scheme performs better than the PDMR scheme. It computes a forwarding table to send data between nodes by maximizing capacity usage based on the maximum flow problem. Further, the MCOP scheme gives better results than PDMR and MPSDN schemes. The scheme selects the path with the largest bandwidth by calculating the weight on that path, which results in fewer QoS violations. However, the proposed approach performs better than the existing schemes in both network topologies. The proposed method considers the residual capacity of the links and allocates the best QoS path that satisfies the bandwidth requirements of user requests. If the proposed method does not consider capacity utilization, it leads to more link congestion and QoS violated flows.



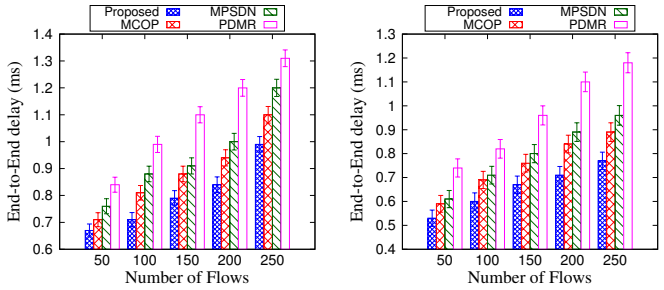
(a) AttMpls Topology (b) Goodnet Topology

Fig. 8: Packet loss in the network with increasing number of flows

3) Packet Loss: We estimated the percentile of packet loss of different traffic flows by a varying number of flows of AttMpls and Goodnet topologies. Fig. 8 illustrates the packet loss with a varying number of flows. It is observed that the proposed scheme reduces the packet loss compared to the benchmark schemes – MCOP, MPSDN, and PDMR. From Fig. 8, we can see that PDMR has the lowest performance, and a large percentage of packets are lost with a rise in flows in both AttMpls and Goodnet topologies. The PDMR scheme results in link congestion and further leads to packet drop in the network.

On the contrary, the packet loss percentage in the MPSDN scheme is lesser than in the PDMR scheme, as the scheduler sends the amount of data into multiple paths by accounting for congestion in the network. Therefore it results in less percentage of the packet drop. In Goodnet topology, the packet drop percentage is higher than the AttMpls topology as the outgoing degree of the number of nodes and the links is lesser than AttMpls, leading to network congestion. However, the MCOP scheme performs better than the PDMR and MPSDN schemes by showing a reduction in the percentile of packet loss. The scheme efficiently utilizes the network resources

while optimizing the link weights. On the other hand, the proposed scheme performs better than benchmark schemes because the scheduler schedules the traffic flows onto multiple paths by considering the link utilization in the network. It avoids overloading the congested paths and thus results in less percentile of packet drop.



(a) AttMpls Topology (b) Goodnet Topology

Fig. 9: End-to-end delay in the network with increasing number of flows

4) End-to-End Delay: Fig. 9 depicts the average delay in routing the traffic with the increasing flows using two network topologies – AttMpls and Goodnet. The proposed approach results in a minor delay in routing traffic flows than benchmark schemes – MCOP, MPSDN, and PDMR. In particular, the proposed method reduces end-to-end delay by 10%, 18%, 36% (with AttMpls topology) and 15%, 21%, 46% (with Goodnet topology) as compared to MCOP, MPSDN, and PDMR schemes, respectively.

From Fig. 9, we observe that the PDMR experiences the lowest reduction in delay with the rise in the number of flows in both AttMpls and Goodnet topology. The link capacity has exceeded, and the scheme does not meet the dynamic requirements of traffic flows. However, the MPSDN scheme suffers from fewer fluctuations and incurs a decline in delay compared to the PDMR method as the scheduler adapts to changes with path characteristics by considering delay and congestion. On the contrary, the MCOP scheme shows a reduction in delay compared to MPSDN and PDMR methods. The scheme uses most of the available bandwidth and avoids load imbalance, resulting in a decrease in end-to-end delay. However, the proposed scheme significantly reduces end-to-end delay compared to existing methods. In the proposed scheme, the controller classifies traffic flows based on their priority, and then the multipath scheduler sends the traffic flows over different paths without overloading the congested route. It directly depends on the smaller percentage of flows that share the path, resulting in less link congestion. Thus, the proposed scheme satisfies the QoS requirements with increased flows.

VI. Practical Applications

This section presents some of use-case scenarios, where the proposed scheme can be used to fulfill application-specific QoS requirements.

- *5G networks:* The emerging applications in 5G are broadly categorized into three aspects – enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (uRLLC), and massive machine-type communications (mMTC) [40]. The emerging eMBB applications, such as virtual reality and online gaming, have high bandwidth requirements of up to 1 Gbps. The proposed scheme is beneficial to support such high bandwidth requirements using the multipath routing while considering their delay requirements.
- *IoT networks:* With the increasing number of IoT applications, the heterogeneity of such applications in terms of QoS requirements is also increasing. Consequently, the network administrators need to consider each application's QoS requirements while routing the data packets in the network. The proposed scheme can be used to meet the diverse QoS requirements of IoT applications.

VII. Conclusions

In this work, we proposed a flow scheduling scheme using multipath technology to balance the network traffic and provide better QoS to the end-users. The main aim is to use the aggregated capacity of multiple paths to transfer data between end-points and provide higher network throughput. SDN takes advantage of a centralized controller that considers the link load to forward the traffic flows and reduce network congestion. The controller uses the K-shortest path algorithm to find the *best* routing paths for traffic flow. We leverage SDN architecture to implement packet scheduling, packet reordering, and path selection to transfer the high volume of data from end-user applications onto multiple paths. Further, we aim to minimize the cost of routing traffic flows in the SDN network. The simulation results have been presented to reflect the effectiveness of the proposed approach to optimize the network throughput, end-to-end delay, and QoS violated flows. In particular, with 250 flows, the proposed approach achieves higher network throughput 7%, 18%, 27% (with AttMpls topology) and 9%, 16%, 25% (with Goodnet topology) as compared to MCOP, MPSDN, and PDMR benchmark schemes, respectively.

In this work, we noticed the flow-rule congestion problem in the presence of large number of applications with diverse QoS requirements. We plan to explore the adaptive flow-rule placement scheme in the presence of large number of applications while considering the limited TCAM available at the SDN switches. Furthermore, the proposed scheme considers the high bandwidth requirements of eMBB applications. However, considering the emerging 5G applications with diverse performance requirements, we plan to include uRLLC and mMTC applications as other use-case scenarios.

References

[1] W. Jiawei, Q. Xiuquan, and J. Chen, "PDMR: Priority-based dynamic multi-path routing algorithm for a software defined network," *IET Commun.*, vol. 13, no. 2, pp. 179–185, 2018.

[2] X. Huang, T. Yuan, and M. Ma, "Utility-optimized flow-level bandwidth allocation in hybrid SDNs," *IEEE Access*, vol. 6, pp. 20 279–20 290, 2018.

[3] P. Kamboj and S. Pal, "QoS in software defined IoT network using blockchain based smart contract," in *Proc. Embedded Networked Sensor Syst.*, New York, USA, Nov. 2019, pp. 430–431.

[4] R. H. Jhaveri, S. V. Ramani, G. Srivastava, T. R. Gadekallu, and V. Aggarwal, "Fault-resilience for bandwidth management in industrial software-defined networks," *IEEE Trans. on Netw. Sci. and Eng.*, vol. 8, no. 4, pp. 3129–3139, 2021.

[5] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "An SDN architecture for time sensitive industrial IoT," *Comput. Netw.*, vol. 186, p. 107739, 2021.

[6] M. Pizzutti and A. E. Schaeffer-Filho, "Adaptive multipath routing based on hybrid data and control plane operation," in *Proc. IEEE INFOCOM*, Paris, France, April 2019, pp. 730–738.

[7] P. Kamboj, S. Pal, and A. Mehra, "A QoS-aware routing based on bandwidth management in software-defined IoT network," in *Proc. MASS*, Denver, USA, Oct. 2021, pp. 579–584.

[8] P. Kamboj and S. Pal, "A policy based framework for quality of service management in software defined networks," *Telecommun. Syst.*, vol. 78, no. 3, pp. 331–349, 2021.

[9] A. Jha, K. K. Singh, K. V. Devi, and V. Manjula, "Reinforcement learning based weighted multipath routing for datacenter networks," *Mater. Today: Proc.*, 2021.

[10] P. Kamboj and S. Pal, "Software-defined networking in data centers," in *Software Defined Internet of Everything*. Springer, 2022, pp. 177–203.

[11] 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 *Proc. IEEE ICC*, Kansas City, USA, May 2018, pp. 1–7.

[12] K. Gao, C. Xu, X. Ji, J. Qin, S. Yang, L. Zhong, and D. Wu, "Freshness-aware age optimization for multipath TCP Over Software Defined Networks," *IEEE Trans. on Netw. Sci. and Eng.*, pp. 1–1, 2021.

[13] W. Zhang, W. Lei, and S. Zhang, "A multipath transport scheme for real-time multimedia services based on software-defined networking and segment routing," *IEEE Access*, vol. 8, pp. 93 962–93 977, 2020.

[14] F. Naeem, G. Srivastava, and M. Tariq, "A software defined network based fuzzy normalized neural adaptive multipath congestion control for the internet of things," *IEEE Trans. on Netw. Sci. and Eng.*, vol. 7, no. 4, pp. 2155–2164, 2020.

[15] D. Banfi, O. Mehani, G. Jourjon, L. Schwaighofer, and R. Holz, "Endpoint-transparent multipath transport with software-defined networks," in *Proc. LCN*, Dubai, UAE, Nov. 2016, pp. 307–315.

[16] W. Wang, W. He, and J. Su, "M2sdn: Achieving multipath and multihoming in data centers with software defined networking," in *Proc. IEEE Int. Symposium on Quality of Service*, Portland, USA, Jun. 2015, pp. 11–20.

[17] Y. Cheng and X. Jia, "NAMP: Network-aware multipathing in software-defined data center networks," *IEEE/ACM Trans. on Netw.*, vol. 28, no. 2, pp. 846–859, 2020.

[18] 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 *Proc. Int. Conf. on Emerg. Netw. Exp. and Technol.*, 2021, pp. 126–140.

[19] T. Zhang, Y. Lei, Q. Zhang, S. Zou, J. Huang, and F. Li, "Fine-grained load balancing with traffic-aware rerouting in datacenter networks," *J. of Cloud Comput.*, vol. 10, no. 1, pp. 1–20, 2021.

[20] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath TCP," in *Proc. Int. Conf. on Advanced Inf. Network. and Appl. Workshops*, Victoria, May 2014, pp. 749–752.

[21] A. Singh, C. Goerg, A. Timm-Giel, M. Scharf, and T.-R. Banniza, "Performance comparison of scheduling algorithms for multipath transfer," in *Proc. IEEE GLOBECOM*, Anaheim, USA, Dec. 2012, pp. 2653–2658.

[22] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Proc. of the Asia Pacific Signal and Information Processing Association Annual Summit and Conf.*, CA, USA, Jan 2012, pp. 1–8.

[23] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Goncalves, R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, "Implementing quality of service for the software defined networking enabled future Internet," in *Proc. IEEE European Workshop on Software Defined Netw.*, Budapest, Hungary, Dec 2014, pp. 49–54.

[24] N. Saha, S. Bera, and S. Misra, "Sway: traffic-aware QoS routing in software-defined IoT," *IEEE Trans. on Emerg. Topics in Comput.*, vol. 16, no. 6, pp. 390 – 401, 2018.

[25] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using SDN," in *Proc. IEEE Real-Time Syst. Symposium*, Paris, France, Feb. 2017, pp. 231–242.

[26] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: An SDN-based multipath QoS solution," *China Commun.*, vol. 12, no. 5, pp. 123–133, 2015.

[27] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Adaptive and reliable multipath provisioning for media transfer in SDN-based overlay networks," *Comput. Commun.*, vol. 106, pp. 107–116, 2017.

[28] C. Lin, Y. Bi, H. Zhao, Z. Liu, S. Jia, and J. Zhu, "DTE-SDN: A dynamic traffic engineering engine for delay-sensitive transfer," *IEEE Internet of Things J.*, vol. 5, no. 6, pp. 5240–5253, 2018.

[29] N. Farrugia, V. Buttigieg, and J. A. Briffa, "A globally optimised multipath routing algorithm using SDN," in *Proc. IEEE Conf. on Innovation in Clouds, Internet and Netw. and Workshops*, Paris, France, Feb. 2018, pp. 1–8.

[30] M. Bagaa, D. L. C. Dutra, T. Taleb, and K. Samdanis, "On sdn-driven network optimization and qos aware routing using multiple paths," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 7, pp. 4700 – 4714, 2020.

[31] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Flexible traffic splitting in openflow networks," *IEEE Trans. on Netw. and Service Manag.*, vol. 13, no. 3, pp. 407–420, 2016.

[32] Q. Wang, G. Shou, Y. Liu, Y. Hu, Z. Guo, and W. Chang, "Implementation of multipath network virtualization with SDN and NFV," *IEEE Access*, vol. 6, pp. 32 460–32 470, 2018.

[33] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[34] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[35] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. on Sel. Areas in Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.

[36] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Netw.*, New York, USA, Oct. 2010, pp. 1–6.

[37] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, 2012.

[38] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE INFOCOM Workshop*, Atlanta, USA, May 2017, pp. 559–564.

[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 Trans. on Netw. and Service Manage.*, vol. 18, no. 4, pp. 4376–4390, 2021.

[40] P. Sarigiannidis, T. Lagkas, S. Bibi, A. Ampatzoglou, and P. Bellavista, "Hybrid 5G optical-wireless SDN-based networks, challenges and open issues," *IET Netw.*, vol. 6, no. 6, pp. 141–148, 2017.