Q-Soft: QoS-aware Traffic Forwarding in Software-Defined Cyber-Physical Systems

Samaresh Bera, *Member, IEEE,* Sudip Misra, *Fellow, IEEE,* Niloy Saha, *Student Member, IEEE,* and Hamid Sharif, *Fellow, IEEE*

***Abstract*—The next-generation cyber-physical systems (CPS) with heterogeneous applications have diverse quality-of-service (QoS) requirements in terms of throughput, end-to-end latency, and packet drop reliability. To meet such diverse QoS requirements, in this paper, we propose a QoS-aware traffic forwarding scheme in software-defined CPS. The proposed scheme is presented as a two-stage optimization framework to minimize the associated costs in traffic forwarding. In the first stage, we aim to minimize the required number of 'candidate' switches for a given network to minimize network deployment costs. In the second stage, we design a comprehensive cost function considering end-to-end delay, flow-rule utilization, and link utilization in the network. Based on the designed cost function, we formulate another optimization problem for optimal traffic forwarding (OTF). As solving OTF is NP-hard, we propose an efficient greedy-heuristic approach to solve the problem while considering application-specific QoS requirements. Further, we propose a packet-tagging method to assist the controller in mitigating rule congestion at the software-defined networking devices, and hence improve the overall network performance. Extensive results show that the proposed scheme minimizes the network delay and QoS-violated flows by up to 50% and 90%, respectively, compared to the state-of-the-art schemes.**

***Index Terms*—Software-defined network, Traffic engineering, Optimization, Packet-tagging, Quality-of-service**

I. Introduction

With the integration of computing, control, and networking technologies, cyber-physical systems (CPS) are expected to play a key role in establishing next-generation *smart* systems. The inherent technology in CPS abstracts the underlying mathematical modeling and software development spanning across multiple domains of computing, control, and networking [1]. Thus, the next generation CPS is expected to include heterogeneous devices in a single platform. However, the integration of heterogeneous devices poses interoperability and networking challenges to the next generation CPS. To address such challenges in a simplified manner, unified and improved software and networking model is required. To this, software-defined networking (SDN) technology is capable of addressing such issues and challenges through the simplified and flexible networking paradigm [2]. Motivated by this, in this work, we focus on the software-defined traffic forwarding in the core network of cyber-physical systems.

The inherent features of SDN technology enable real-time programmability of networking devices while separating the *control-* and *data-* planes from the devices. Thus, the complexity involved in network management in the SDN-enabled network is reduced significantly compared to that of the traditional networks. In SDN-based traffic forwarding, on receiving a new data-packet, a switch sends a meta-data of the received data-packet as `Packet-In` message to the SDN controller. Then, the switch takes action (forward or drop) on the received data-packet according to the flow-rule defined by the controller. However, due to limited memory at the switches, an existing flow-rule corresponding to active traffic flows is removed to accommodate a new flow-rule insertion. This removal triggers repeated requests at the controller by data-packets corresponding to the ongoing flow, which increases delay and controller overhead. Consequently, it entails degraded quality-of-service (QoS) in terms of throughput, end-to-end delay, and packet drop. Recently, Qiao et al. [3] proposed a random packet forwarding scheme on rule congestion at SDN switch without generating `Packet-In` to the controller to alleviate this issue. Figure 1 summarizes the scheme proposed in [3] that uses randomization to reduce the control plane overhead. As depicted in Figure 1, the scheme

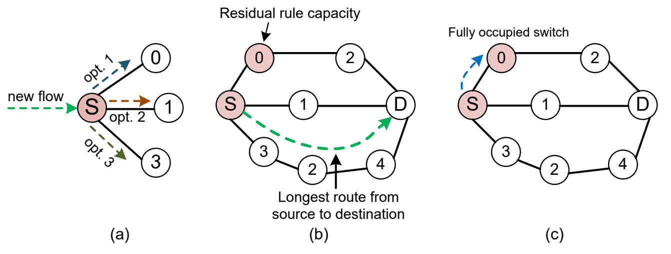


Fig. 1: The randomly selected port for traffic forwarding lead to – a) longest path in traffic forwarding; and/or b) another fully utilized switch

has the following limitations as follows: a) as the outgoing port for an incoming packet is chosen randomly, the packet may be forwarded through the longest path, instead of the shortest one; and b) the randomly chosen switch may also be fully utilized. In such a case, the packet is further forwarded

S. Bera and N. Saha were with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, 721302, India, where the work was done.

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

H. Sharif is with the Electrical and Computer Engineering, University of Nebraska–Lincoln, NE 68588, USA, Email: hsharif@unl.edu.

A preliminary version of this work has been presented in the IEEE ICC workshop 2018, Kansas City, USA (May 20–24, 2018), DOI: 10.1109/ICCW.2018.8403550. The authors gratefully acknowledge the funding support received from SERB/IMPRINT-II (Sanction letter no SERB/F/12680/2018-2019; IMP/2018/000451, Dt. 25-03-2019) for executing parts of this work.

to a randomly selected switch. As a result, the controller is unaware of such traffic flow in the network as `Packet-In` is not generated. Moreover, individual QoS requirements of the flows may be violated due to the randomized forwarding. This problem is much more worsening in the presence of a huge number of heterogeneous flows that is obvious in a CPS.

To address the above-mentioned issues, we propose a QoS-aware traffic engineering scheme for software-defined CPS that a) focuses on efficient traffic forwarding considering the QoS requirements of applications, and b) takes into account the rule-utilization of switches in the network. The proposed scheme consists of two phases. In the first stage, we formulate an integer linear program (ILP) to reduce the number of switches facilitated with hybrid flow-rule capacity [4], termed as *candidate switches*, for a given network to minimize the network deployment cost. In the second stage, we design a comprehensive cost function that takes into account the end-to-end delay, flow-rule utilization, and link capacity utilization in the network. Based on the designed cost function, we formulate another optimization problem for optimal traffic forwarding (OTF) for the incoming flows to minimize the forwarding cost. As solving OTF is NP-hard, we propose an efficient greedy-heuristic approach to solve the problem considering the application-specific QoS requirements. Furthermore, we propose a lightweight packet-tagging approach to proactively inform the SDN controller about rule-congestion at SDN switches, which is further used to make forwarding decisions in the future. Experiment results show that network performance in terms of network delay, packet loss, and QoS-violated flows can be improved using the proposed scheme compared to the benchmark schemes. We identify two CPS use-case scenarios and discuss the potential of the proposed scheme in addressing the QoS requirements of the systems. In brief, the contributions in this work are as follows.

- We formulate an optimization problem to minimize the number of hybrid switches (termed as candidate switches) to minimize the network deployment cost of a software-defined CPS.
- We formulate another optimization problem that takes into account the placement of candidate switches and reduces the associated forwarding cost in terms of end-to-end delay, flow-rule capacity, and link capacity utilization. To efficiently solve the problem within time-bounds, we propose an efficient greedy-heuristic approach for selecting forwarding paths, while satisfying the QoS requirements of the applications.
- We propose a packet-tagging approach to explicitly notify the SDN controller about flow-rule congestion so that the controller can take adequate actions.
- The performance is evaluated using the Mininet network emulator and the POX SDN controller, and we show that the network performance can be enhanced significantly compared to the benchmark schemes — `SPD`, `MRC`, `RFS`, and `Sway`.

We organize the rest of the paper as follows. Section II discusses the existing schemes while identifying the limitations. Section III presents the proposed dynamic traffic engineering scheme in software-defined CPS. Section IV reports the experiment results. Two CPS use-case scenarios are presented in Section V to show the suitability of the proposed scheme in practical scenarios. Finally, in Section VI, we conclude the paper while highlighting the limitations of the proposed scheme and some future research directions.

II. Related Work

In this section, we review the existing traffic engineering schemes in SDN and CPS. Recent studies [2], [5] showed that software-defined networking aspects are useful to address the issues and challenges in building CPS. Further, several schemes [3], [4], [6]–[15] also focused on traffic forwarding in SDN. For example, the benefits of SDN-based traffic engineering are studied in [7]. With the presence of a global-view of the network, the SDN controller can take optimal decisions on traffic forwarding. To deploy SDN-based traffic forwarding, traditional networking devices need to be replaced by SDN switches. However, it is cost-expensive to migrate the entire network with SDN switches. Caria et al. [6] proposed an algorithm to determine the optimal number of SDN switches considering the associated deployment cost. Thus, the authors showed that the network capacity up-gradation can be minimized, which, in turn, minimizes the deployment cost.

TABLE I: Summary of existing works

Work	Contribution(s)	Remarks
Qiao et al. [3]	Randomized traffic forwarding on detecting rule congestion	May lead to QoS violation of flows
Katta et al. [4]	Use of hybrid switches for rule placement	Increased CAPEX and may lead to increased delay due to hardware-software dependency
Saha et al. [14]	Traffic-aware QoS routing for IoT application	Considered unlimited flow-rule space available at switches
Bhatia et al. [8]	Segment routing to avoid rule congestion	May lead to QoS violation and security threat to the network

Segment routing is another important aspect in traffic engineering by simplifying the forwarding mechanisms. In particular, a source node can specify a unicast forwarding path using the segment routing rather than specifying the shortest path, through which the packet will traverse. Authors in [8]–[10] proposed a segment routing-based traffic engineering scheme in SDN-enabled networks in which an architecture for segment routing is also presented. In segment routing, the SDN switches forward a packet to its next-hop switch without sending `Packet-In` to the controller. Thus, frequent rule placement at the switches can be avoided. Such an approach is useful while the rule capacity of an SDN switch is nearly/completely utilized. However, as mentioned in Section I, such a forwarding scheme may lead to an increased delay in traffic forwarding.

Concurrently, Qiao et al. [3] proposed a way-point routing scheme in the presence of congested switches with active flow-rules. In such a scenario, the congested switch forwards the traffic to a randomly selected neighbor without generating `Packet-In` to the controller. Consequently, the flow-

TABLE II: List of symbols

Symbol	Description
$\mathcal{S}$	Set of switches in the network
$\mathcal{L}$	Set of links between switches
$\mathcal{C}$	Set of candidate switches
$\mathcal{F}$	Set of flows
R_i^{max}	Rule capacity of a switch $i \in \mathcal{S}$
R_i^{util}	Utilized rule capacity of a switch $i \in \mathcal{S}$
$L_{i,j}^{util}$	Utilized link capacity of a link $(i,j) \in \mathcal{L}$
$L_{i,j}^{max}$	Link capacity of a link $(i,j) \in \mathcal{L}$
w_j	integer variable
$\beta_{i,j}$	binary variable
$d_{i,j}$	Delay of a link $(i,j) \in \mathcal{L}$
d_f	Delay requirement of a flow $f \in \mathcal{F}$
$\Phi_{i,j}^f$	Cost for routing a flow f over link (i,j)

rules at the switch for active flows are not replaced by the new traffic. Thus, message overhead for rule placement is avoided. Recently, Saha et al. [14] proposed a QoS-aware routing scheme in SDN while considering the requirements of IoT applications. They categorized the IoT flows from two aspects — delay-sensitive and loss-sensitive. Consequently, the authors proposed a greedy heuristic to forward such traffic, while considering the individual requirements of the IoT flows.

Table I presents a detailed comparison of the existing works that focused on QoS-aware traffic forwarding in SDN. We see that the existing schemes may not be suitable for differentiated service provisioning in a software-defined CPS integrated with heterogeneous devices and applications.

III. QSoft: Quality-Aware Traffic Forwarding

Let there be an SDN-enabled network in a CPS denoted as a directed graph $\mathcal{G} = (\mathcal{S}, \mathcal{L})$, where $\mathcal{S}$ and $\mathcal{L}$ denote the set of SDN switches and set of links between the switches, respectively. Further, $\mathcal{S}$ and $\mathcal{L}$ are denoted as follows: $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ and $\mathcal{L} = \{(i,j) | (i,j) \subset \mathcal{S} \times \mathcal{S}, i \neq j\}$. The list of symbols used in this work is shown in Table II.

Candidate Switch: We denote a switch as hybrid or candidate switch (CS) if it has the provision of both hardware and software packet processing facilities. Therefore, such a combination of software and hardware processing provides more flexibility and large rule-space [4]. Although hybrid switches offer increased rule space, they may be relatively expensive. Consequently, the objective is to minimize the number of candidate switches in the network without degrading the network performance. Figure 2 presents the schematic view of the architecture of a CS comprising of both hardware and software facilities. The software part can be placed at the switch itself or in a remote place depending on the network deployment. Further, *candidate switch master* acts as an agent and controls the flow-rule placement and `Packet-In` messages at both the software and hardware switches. We limit our discussion on the candidate switch architecture in this paper (details can be found in [4]).

Controller architecture: Figure 3 presents the proposed SDN controller architecture with data-plane. The *Switch Manager* is connected to the *Candidate Switch Master* and the general switches present in the network. The *`Packet-In` Handler* module collects the OpenFlow `Packet-In` messages sent from the data-plane. The *Q-Soft* module determines the forwarding path for traffic forwarding while considering the application-specific requirements (proposed in Section III-B). The *Flow-Rule Manager* deploys the flow-rules according to the forwarding path decided by the *Q-Soft* module. The *Packet-Tag Identifier* module is used to identify the rule-congestion in the network (proposed in Section III-C). The default active-probing method is used to collect network statistics.

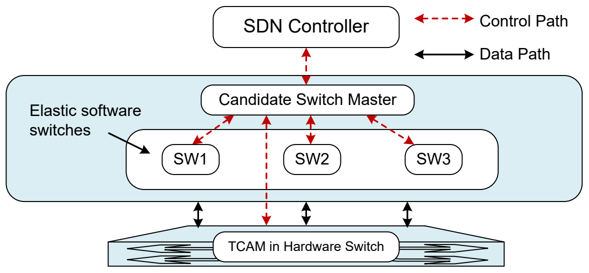


Fig. 2: Candidate switch architecture

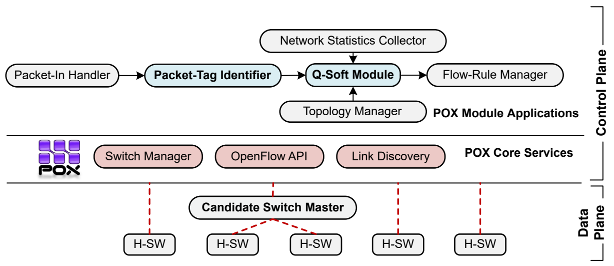


Fig. 3: Controller architecture used in this work

A. Candidate Switch Selection

The candidate switch (CS) provides more flexibility in packet processing in the network. Therefore, the general purpose switches need to be replaced by the CSs in order to have increased network flexibility. However, cost of such hybrid switches is more compared to the general switches. Therefore, for a given network $\mathcal{G}(\mathcal{S}, \mathcal{L})$, the objective is to minimize the number of switches from $\mathcal{S}$ that need to be designed as CS. We denote the set of CS as $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$, where $k \in [1, |\mathcal{S}|]$ and $\mathcal{C} \subseteq \mathcal{S}$. Mathematically,

$$\text{Minimize} \sum_{j=1}^{|\mathcal{S}|} w_j$$

subject to

$$\sum_j \beta_{i,j} \geq 1, \forall i \in \mathcal{S}, \tag{1}$$

$$\beta_{i,j} \leq w_j, \forall i \in \mathcal{S} \text{ and } j \in \mathcal{C}, \tag{2}$$

where $w_j \in Z_{>=0}$ is an integer variable and it captures the number of switches selected as CSs. Equation (1) represents that every switch has at least one CS in its one-hop distance. Equation (1) is used as a trade-off between the cost of CS and the control overhead in terms of `Packet-In` in the network. Therefore, we consider the distance between a switch and CS as one-hop in this work, but it is user-defined. The binary variable $\beta_{i,j}$ denotes the availability of a CS, $j \in \mathcal{C}$, from another switch, $i \in \mathcal{S}$. Mathematically,

$$\beta_{i,j} = \begin{cases} 1, & \text{if there exists a link between} \\ & \text{switch, } i \in \mathcal{S}, \text{ and CS, } j \in \mathcal{C} \\ 0, & \text{Otherwise.} \end{cases} \tag{3}$$

Equation (2) restricts the number of switches to be selected as CS. The formulated optimization problem is an integer linear program (ILP). The optimization problem is solved using the GLPK solver (https://www.gnu.org/software/glpk/). It is noteworthy that this problem is solved once during the implementation stage and does not change dynamically over time.

B. *Optimal Traffic Forwarding (OTF)*

As discussed in Section I, typically, a CPS is comprised of heterogeneous devices. Therefore, the network carries heterogeneous traffic generated from applications that can be event-driven (e.g., smart alarm system), low-rate (e.g., precision agriculture), and have different bandwidth requirements. Moreover, packet loss- and delay-guaranteed data delivery are another two important quality of service (QoS) requirements while forwarding traffic through the network [14]. Consequently, deciding optimal policies considering multiple metrics poses significant challenges to the controller while forwarding incoming traffic in the network. Consequently, we design a cost function with the following properties.

$$\frac{\partial \Phi(R,L,d)}{\partial R} \geq 0, \frac{\partial \Phi(R,L,d)}{\partial L} \geq 0, \text{ and } \frac{\partial \Phi(R,L,d)}{\partial d} \geq 0,$$

where R is associated with rule-space utilization at the forwarding switch. L and d denote link utilization and link delay between two forwarding switches, respectively. We introduce a binary variable $x^f_{i,j}$ as follows:

$$\sum_{j\in N(i)} x^f_{i,j} = \begin{cases} 1, & \text{if switch } i \text{ is included to forward flow } f \\ 0, & \text{Otherwise,} \end{cases}$$

where $j \in N(i)$ denotes the neighboring switches of switch j. Accordingly, the link- and rule-space utilization are calculated as:

$$L^{util}_{i,j} = \sum_{f\in\mathcal{F}} b^f x^f_{i,j}, \quad \text{and} \quad R^{util}_i = \sum_{f\in\mathcal{F}} \sum_{j\in N(i)} x^f_{i,j}, \tag{4}$$

where b^f is bandwidth requirement of flow f.

Consequently, we formulate the cost function considering the above-mentioned parameters as follows:

$$\Phi^f_{i,j}(R,L,d) = \alpha \frac{R^{util}_i}{R^{cap}_i} + \beta \frac{L^{util}_{i,j}}{L^{cap}_{i,j}} + \gamma \frac{d_{i,j}}{d^f}, \; i,j \in \mathcal{S} \wedge i \neq j \tag{5}$$

where α, β, and γ are user-defined constants to consider the relative importance between the rule-utilization, link-utilization, and link delay, respectively. R^{util}_i denotes the rule-space utilization at switch i. $L^{util}_{i,j}$ and $d_{i,j}$ denote the link utilization and link delay between the switches i and j, where $i \neq j$. Further, R^{cap}_i presents the rule capacity of the switch i. $L^{cap}_{i,j}$ is the link capacity between the switches i and j. The term d^f denotes the allowable delay associated with flow f. Thus, for a given network and cost of a link to forward a flow, the objective is to reduce the forwarding cost for all flows in the network while considering network capacity and QoS requirements of flows. Mathematically,

$$\text{Minimize} \sum_{f\in\mathcal{F}} \sum_{e\in\mathcal{L}} \Phi^f_{i,j} x^f_{i,j}$$

subject to

$$x^f_{i,j} \in \{0,1\}, \; \forall i,j \in \mathcal{S} \tag{6}$$

$$L^{util}_{i,j} \leq L^{cap}_{i,j}, \; \forall i,j \in \mathcal{S} \tag{7}$$

$$R^{util}_i \leq R^{cap}_i, \; \forall i \in \mathcal{S} \tag{8}$$

$$\sum d_{i,j} x^f_{i,j} \leq d^f, \; \forall i,j \in \mathcal{S} \tag{9}$$

where $\Phi^f_{i,j}$ denotes the cost of a link (i,j) while forwarding a flow f, and $x^f_{i,j}$ denotes whether the link (i,j) is selected to forward the flow f. Therefore, our objective is to minimize the overall cost for forwarding the flows in the network. Equation (6) captures the selection of the link (i,j) to forward the flow f. Equation (7) ensures that the link utilization must be less than or equal to the total capacity of the link. Equation (8) preserves the rule capacity constraint of a switch. Finally, Equation (9) denotes that the total delay incurred by the flow over the links (i,j) from source to destination is within the allowable delay associated with the flow.

1) Greedy Heuristic for Traffic Forwarding: The above-formulated ILP is NP-hard in general [16]. Therefore, we propose a greedy-heuristic approach to forward the traffic in the network. Algorithm 1 presents the proposed greedy algorithm for traffic forwarding while considering the associated cost presented in Equation (5). For a given flow, we use Yen's K-Shortest path [17] to get all possible paths that satisfy the flow requirements, as presented in 3. In Step 4, flow-requirement is checked to maintain QoS in traffic forwarding. Finally, the path with minimum cost is chosen from the list of qualified paths, which is presented in Step 8. It is noteworthy that the proposed algorithm forwards the flows one-by-one in the network in an online fashion. The time complexity of the proposed greedy approach (presented in Algorithm 1) is analyzed. The *for* loop in Step 1 runs $|\mathcal{F}|$ times for total number of flows. Further, we use Yen's K-shortest path algorithm (refer to Step 5), which is the most time-expensive operation in the proposed greedy approach. The time complexity to get K-shortest path using Yen's algorithm is $O(K|\mathcal{S}|(|\mathcal{L}| + |\mathcal{S}| \log |\mathcal{S}|))$. Therefore, the total time complexity of the proposed greedy algorithm is $O((|\mathcal{F}|) \times (K|\mathcal{S}|(|\mathcal{L}| + |\mathcal{S}| \log |\mathcal{S}|))$. It is noteworthy that the proposed greedy approach runs in polynomial time.

To show the efficacy of the proposed heuristic in solving the traffic forwarding problem, we present empirical results comparing the heuristic to the ILP formulated in Section III-B. We create a network with 12 nodes using the Barabasi-Albert scale-free topology. Flows are generated using the values presented in performance evaluation (refer to Table III). Figure 4

Algorithm 1 Greedy algorithm for traffic forwarding (OTF)

```
Inputs:
    Graph, $\mathcal{G}$; Flow $f$ with requirements; Maximum rules at
    a switch $R^{cap}$; Values for constants $\alpha$, $\beta$, and $\gamma$
Output:
    Forwarding path $p^f$ with associated cost $\Phi^f$ on which flow
    $f$ can be routed
 1: Compute path for flow $f$ using GET-PATH($s, t, f$)
 2: function GET-PATH($s$, $t$, $f$)
 3:     for path in K-SHORTEST-PATHS($s$, $t$) do
 4:         if CHECK-REQUIREMENT($path$, $f$) then
 5:             Calculate cost $\Phi^f$ according to Equation (5)
 6:             $\Theta^f \leftarrow (path, \Phi^f)$
 7:     if $\Theta^f$ then
 8:         Choose the minimum cost path $p^f$ from $\Theta^f$
 9:         return path $p^f$
10:     else
11:         Flow $f$ is considered as QoS-violated
    ▷       No QoS path found
12: function CHECK-REQUIREMENT($path$, $f$)
13:     if CHECK-LINK-DELAY and CHECK-RULE-CAPACITY
         and CHECK-LINK-CAPACITY then
14:         return True
15:     else
16:         return False
```

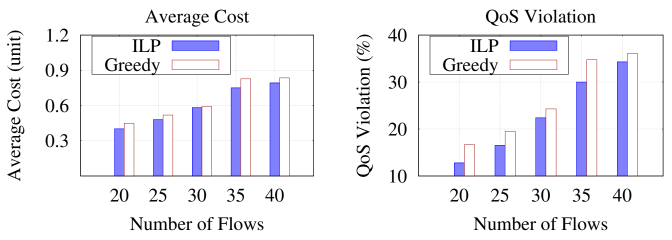
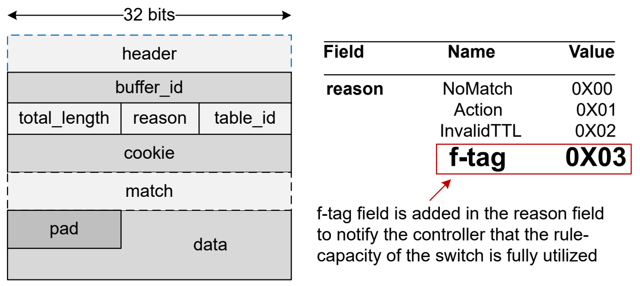


Fig. 4: Comparison between the optimal solution and the proposed greedy approach

represents the performance comparison between the ILP and the proposed greedy approach for traffic forwarding. We see that the proposed scheme gives competitive results to the ILP in terms of cost and QoS-violated flows. Further, it is also noteworthy that the computation time increases exponentially in the case of ILP with an increasing number of flows in the network.

C. Tagging the Packet-In Message

Typically, in SDN, rule congestion leads to frequent rule replacement at the switches, which, in turn, increases the end-to-end delay in packet delivery. A modified version of the OpenFlow protocol is used to notify the SDN controller about the rule congestion. Each switch uses the modified `Packet-In` message on detecting rule congestion, as shown in Figure 5. We follow the OpenFlow specification [18] to achieve this. In addition to the standard fields, another **name** is used as **f-tag** under the **reason** field, where the **value** of the **f-tag** is set to **0X03** to notify the controller about


32 bits
header
buffer_id
total_length
reason
table_id
cookie
match
pad
data
f-tag field is added in the reason field to notify the controller that the rule-capacity of the switch is fully utilized

Field	Name	Value
reason	NoMatch	0X00
	Action	0X01
	InvalidTTL	0X02
	f-tag	**0X03**

Fig. 5: Modified `Packet-In` message used in packet-tagging method

TABLE III: Simulation Parameters

Parameter	Value
Network Topology	AttMpls and Goodnet [20]
Number of switches	25 (AttMpls) and 17 (Goodnet)
Number of links	57 (AttMpls) and 31 (Goodnet)
Number of flows	100 – 300
Bandwidth requirement	0.20 – 0.40 kbps
Packet size	94 – 699 bytes [21]
Active volume	142 – 27716 bytes [21]
Mean traffic rate	562 – 516,540 bps [21]
Active flow time	1 – 34 s [21]
[α, β, γ]	[0.33, 0.33, 0.33]

rule congestion at the switch. Consequently, on receiving the rule congestion notification, the controller takes an adequate decision to alleviate the congested switch. We believe that the packet-tagging method can be easily integrated into the existing SDN systems that utilize the OpenFlow protocol.

IV. Performance Evaluation

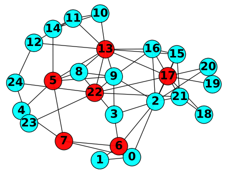
The performance of the proposed scheme, called `Q-Soft`, is evaluated using the POX controller and the Mininet network emulator [19]. We consider two real-life network topologies — AttMpls and Goodnet — from the Internet topology Zoo [20] to consider both dense and relatively sparse network topologies. To generate heterogeneous traffic with diverse QoS requirements, we used D-ITG traffic generator according to the properties mentioned in [21]. The experiments were conducted in a Google Cloud (https://cloud.google.com/) instance configured with Intel Skylake processor and 7.5GB RAM. Table III presents the parameters and their values used to evaluate the performance.

We use different performance metrics such as end-to-end delay, average packet drop, throughput, number of `Packet-In` messages, and QoS-violated flows. The results for the end-to-end delay, packet drop, and throughput are measured using the utilities of D-ITG available in the Mininet network emulator. On the other hand, the number of `Packet-In` and QoS-violated flows in traffic forwarding are measured at the controller-end according to the `Packet-In` handler and Q-Soft module. Further, we compare the proposed scheme, `Q-Soft`, with the existing traffic forwarding schemes — shortest path delay (`SPD`) [22], minimum occupied rule capacity (`MRC`) (as discussed in [23]), randomized forwarding scheme (`RFS`) [3], and QoS-routing scheme (`Sway`) [14]. In `SPD`, the traffic is forwarded through the path which incurs less delay. On the other hand, in MRC, traffic is forwarded through the switches in which the minimum number of flow-rules is installed. Further, in the case of RFS, a switch forwards incoming traffic randomly to one of its outgoing ports without generating Packet-In to the controller when its rule-capacity is fully utilized. However, when the rule-capacity is under-utilized, RFS follows the standard OSPF-based traffic forwarding approach. In Sway, the SDN controller installs flow-rules based on the type of applications – delay-sensitive and loss-sensitive.

A. Results and Discussion

We analyze the obtained results and compared them with the existing scheme as follows.

1) Candidate Switch Selection: As mentioned in Section III-A, the proposed scheme aims to select the optimal number of candidate switches in the network. Consequently, we determine the minimum number of candidate switches in the network by solving the ILP, as mentioned in Section III-A. Figure 6 presents the AttMpls and Goodnet network topologies with candidate switches in red color. From the figure, we see that a small number of switches are required to be associated with software switches for both the network topologies. Accordingly, we set up the network and conduct the experiment to obtain results related to network performance. It is noteworthy that the software switches are placed locally in the experiment.


(a) AttMpls topology

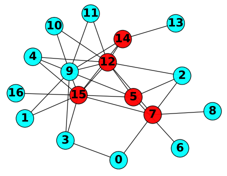

(b) Goodnet topology

Fig. 6: Network topology with candidate switches

2) End-to-End Delay: Delivery of the flows within the desired time-bound is a crucial factor. In the proposed scheme, Q-Soft, we aim to minimize the overall end-to-end delay in traffic forwarding. Figure 7 presents the average packet delivery delay with varying number of flows for AttMpls and Goodnet network topologies. We see that Q-Soft is capable of minimizing the end-to-end delay by 34%, 41%, 50%, and 40% compared to the SPD, MRC, RFS, and Sway schemes, respectively. In the case of Q-Soft, the controller determines the forwarding path in which incoming traffic can be forwarded to the destination while considering associated delay-bound. Further, the proposed cost-based path selection mechanism avoids congested switches in the network to minimize the flow-setup delay. Therefore, the overall delay incurred by the flows is minimized using the proposed scheme, Q-Soft. In contrast, SPD chooses the links with a minimum delay to forward the traffic to the destination. Therefore, for a given source-destination pair, the same path is chosen irrespective of the characteristic of the flows. As a result, the switches in the path are congested and flow-rules are required to be replaced with new ones upon receiving new flows, which, in turn, increases

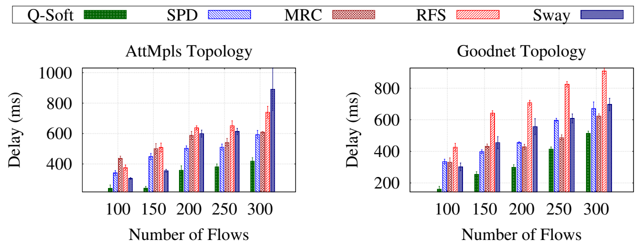


Fig. 7: Average delay using different flows

the flow-setup delay. In MRC, a path consists of switches that are minimally occupied. Therefore, links with higher delay may be selected, which, in turn, increases the packet delivery delay. On the other hand, in RFS, due to random selection of outgoing ports upon full utilization on flow-rule space, the end-to-end delay increases in the network. Finally, in Sway, the delay is not taken into account while forwarding loss-sensitive flows, which, in turn, increases the average network delay compared to Q-Soft. It is also noteworthy that Sway was designed for low-rate traffic. Therefore, the average delay using Sway suddenly increases with a large number of flows in the network.

It is noteworthy that due to the sparse nature of the Goodnet topology, forwarding paths for the delay- and loss-sensitive flows are almost similar for a given pair of source and destination. Therefore, the average delay in the Goodnet network topology is relatively lower than the AttMpls topology. However, the proposed scheme, Q-Soft, always outperforms the existing schemes — SPD, MRC, RFS, and Sway — in terms of end-to-end delay. The delay increases with an increasing number of flows in the network. This is due to more number of Packet-In, increased rule-capacity utilization, and more number of queued packets at the switches. However, it is always better to use the proposed scheme than the existing schemes.

3) Packet Drop: Concurrent to the delay, flows are also loss-sensitive, which necessitates the minimization of packet drop in the network. Figure 8 presents the packet drop that occurred in the network. From the figures, it is evident that Q-Soft is capable of minimizing the packet drop by 40%, 45%, 57% and 54% (AttMpls topology), and 44%, 35%, 63% and 45% (Goodnet topology) compared to SPD, MRC, RFS, and Sway schemes, respectively. In Q-Soft, the packets are forwarded considering the residual link bandwidth, rule capacity at the switches, and link delay. Therefore, congestion at a particular link or switch is avoided using the proposed scheme. Hence, a fewer number of packets are queued at the switches, which, in turn, reduces the number of packet drops. In contrast, in SPD, the number of queued packets is increased due to the shortest-path delay mechanism. This leads to increased packet drop. Similarly, in MRC, the link delay and bandwidth are not considered while forwarding incoming packets. Hence, more packets are dropped due to insufficient link capacity to forward the incoming packets. On the other hand, in RFS, packets are forwarded through wrong outgoing ports due to a random forwarding scheme when

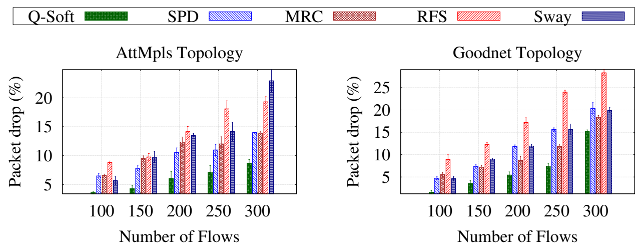


Fig. 8: Average packet drop using different flows

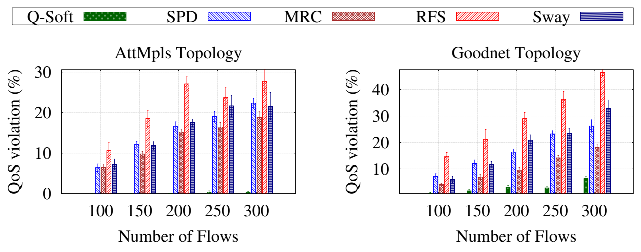


Fig. 10: QoS violation using different flows

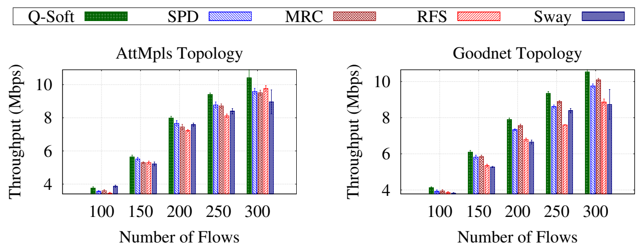


Fig. 9: Average throughput using different flows

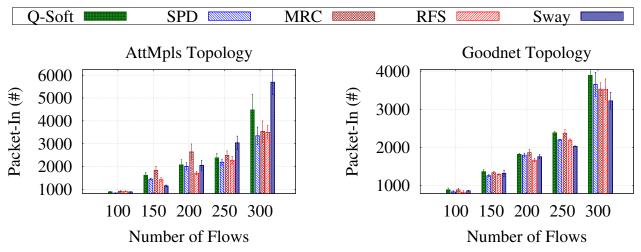


Fig. 11: Number of Packet-In using different flows

the rule capacity is fully utilized. Consequently, the packets forwarded to the wrong outgoing ports are not able to reach the destination. This leads to more packet drops in the network. In `Sway`, the loss is not taken into account while forwarding delay-sensitive flows in the network and vice-versa, which, in turn, increases the average packet drop. Further, the packet drop increases with an increasing number of flows in the network due to the resource-constrained nature of the network. However, the proposed scheme always performs better than the existing schemes.

4) Network Throughput: Figure 9 presents throughput obtained using the proposed and existing schemes. It is observed that the proposed scheme, `Q-Soft`, provides higher throughput compared to the existing schemes — `SPD`, `MRC`, `RFS`, and `Sway`. In `Q-Soft`, the cost-based traffic forwarding scheme ensures the packets to be forwarded in the network for which link utilization is less. Consequently, the throughput using `Q-Soft` is always better than the existing schemes for both the network topologies. Further, in `Sway`, bandwidth utilization was not considered while forwarding traffic in the network. This leads to decreased network throughput although loss and delay requirements for loss- and delay-sensitive applications are considered, respectively.

5) QoS-violated Flows: We present the QoS-violated flows in the network with the different number of flows, as depicted in Figure 10. We check the QoS criteria of the forwarded flows at the controller-end. It is evident from the figure that `Q-Soft` is capable of reducing the QoS-violated flows by 82%, 72%, 89%, and 84% compared to `SPD`, `MRC`, `RFS`, and `Sway` schemes, respectively. In `SPD` and `MRC`, due to insufficient link capacity or rule capacity, required QoS cannot be ensured. Similarly, due to the randomly chosen outgoing ports, packets experience a higher delay (refer to Figure 7), which, in turn, increases the percentage of QoS-violated flows in the network. Further, in the case of high-rate traffic in the network, the percentage of QoS-violated flows using `Sway` is also increased compared to `Q-Soft`.

6) Number of Packet-In: Finally, we present the number of `Packet-In` messages to show the network control overhead in Figure 11. It is observed that the proposed scheme, `Q-Soft`, incurs moderate number of `Packet-In` compared to the existing schemes — `SPD`, `MRC`, `RFS`, and `Sway`. In `Q-Soft`, after receiving a `Packet-In`, the controller computes the optimal path for traffic forwarding and places the flow-rule at the switch, as mentioned in Algorithm 1. During the path computation, the switch keeps on sending the `Packet-In` associated with the same flow until a matched rule is found at the flow-table. Due to this reason, the number of `Packet-In` increases using the proposed scheme. This is also applicable to `SPD` and `MRC` schemes. However, once a flow-rule associated with the flow is installed at the switch, all the queued packets (related to the same flow) are processed. On the other hand, in `RFS`, a switch forwards the traffic to a randomly selected outgoing port without generating `Packet-In` on rule congestion, which, in turn, reduces the network control overhead. Similar to `MRC` and `SPD`, due to the absence of software switch support, more number of `Packet-In` is generated in the `Sway` scheme.

To summarize, we see that the proposed scheme, `Q-Soft`, improves the network performance significantly compared to the existing schemes — `SPD`, `MRC`, `RFS`, and `Sway`, while preserving required QoS requirements of flows.

V. Use-Case Scenario: Practical Aspects

We briefly discuss two CPS scenarios in which the proposed SDN-based traffic engineering would be useful to enhance the network performance while fulfilling the requirements of the flows.

• *Traffic forwarding in SDN-based smart grid system*: In a smart grid, consumers' energy consumption data is reported to meter data management systems (MDMS). Further, the collected information at the MDMS is forwarded to the data center network through the backbone network for further processing and *intelligent* decision making in energy management. Thus, some applications, such as monitoring real-time energy consumption and blackouts, require guaranteed data delivery in real-time, i.e., the applications are delay-sensitive. Whereas a few other applications, such as billing and making business policy, require the information to be delivered with a minimum loss, i.e., the applications are loss-sensitive. In the proposed scheme, we aimed to address such issues while forwarding traffic in the network. Moreover, experiment results show that the proposed scheme is beneficial to minimize network delay and loss while ensuring QoS-guaranteed data delivery. Therefore, we believe that the proposed scheme would be beneficial to address the fundamental issues present in the smart grid communication network.

- *QoS-guaranteed traffic forwarding in SDN-based healthcare systems*: Another important CPS system is healthcare systems. In a healthcare system, major information is delay-sensitive, hence, it requires time-bound data delivery. The proposed scheme is capable of addressing this issue by employing *optimal* traffic forwarding mechanism. Further, the experiment results show that end-to-end delay and QoS violated flows can be reduced significantly using the proposed scheme.

VI. Conclusion

In this paper, we proposed a dynamic traffic engineering scheme intending to maximize overall network performance. The proposed scheme consists of two phases — candidate switch selection and optimal path selection. In the candidate switch selection phase, we determined the set of SDN switches required to be associated with software switches to minimize rule congestion. In the optimal path selection phase, we formulated a cost function for traffic forwarding, so that overall cost is minimized. We used the Mininet emulator and POX SDN controller to evaluate the performance of the proposed scheme. It is observed that the proposed scheme enhances the network performance compared to the existing schemes.

We observed that additional `Packet-In` messages are generated during the path computation phase which leads to an increased number of `Packet-In` using the proposed scheme. This may be addressed by improved data plane packet processing techniques such as P4. Further, the performance of the proposed schemes depends on the network topology, flow-rule capacity of the candidate switches, and network resources. We plan to address these limitations of the proposed scheme as a future extension of this work.

References

[1] K.-D. Kim and P. R. Kumar, "Cyber-Physical Systems: A Perspective at the Centennial," *Proc. IEEE*, vol. 100, pp. 1287–1308, May 2012.

[2] E. Molina and E. Jacob, "Software-defined networking in cyber-physical systems: A survey," *Computers and Electrical Engineering*, vol. 66, 2018.

[3] S. Qiao, C. Hu, X. Guan, and J. Zou, "Taming the Flow Table Overflow in OpenFlow Switch," in *Proc. of the ACM SIGCOMM*, Florianopolis, Brazil, Aug. 2016, pp. 591–592.

[4] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks," in *Proc. of the Symposium on SDN Research (SOSR)*, no. 6, CA, USA, Mar. 2016.

[5] S. Bera, S. Misra, and N. Saha, "DynamiTE: Dynamic Traffic Engineering inSoftware-Defined Cyber Physical Systems," in *Proc. of the ICC Workshop*, no. 6, Kansas City, USA, 2018.

[6] M. Caria, A. Jukan, and M. Hoffmann, "A performance study of network migration to SDN-enabled Traffic Engineering," in *Proc. of IEEE GLOBECOM*, Dec. 2013, pp. 1391–1396.

[7] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic Engineering in Software Defined Networks," in *Proc. of the IEEE INFOCOM*, 2013, pp. 1–9.

[8] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. of the IEEE INFOCOM*, 657–665, Apr–May 2015.

[9] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *Proc. of IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[10] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano, "Traffic Engineering with Segment Routing: SDN-Based Architectural Design and Open Source Implementation," in *Proc. of European Workshop on Software Defined Networks (EWSDN)*, Nov. 2015, pp. 111–112.

[11] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. of the ACM SIGCOMM workshop on HotSDN*, Hong-Kong, China, Aug. 2013, pp. 19–24.

[12] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in *Proc. of the ACM SIGCOMM*, Hong-Kong, China, Aug. 2013, pp. 27–39.

[13] S. Bera, S. Misra, and M. S. Obaidat, "Mobility-Aware Flow-Table Implementation in Software-Defined IoT," in *Proc. of the IEEE GLOBECOM*, Dec. 2016, pp. 1–6.

[14] N. Saha, S. Bera, and S. Misra, "Sway: Traffic-Aware QoS Routing in Software-Defined IoT," *IEEE Trans. Emerg. Topics Comput.*, 2018, DOI: 10.1109/TETC.2018.2847296.

[15] A. Kushwaha, S. Sharma, N. Bazard, A. Gumaste, and B. Mukherjee, "Design, Analysis, and a Terabit Implementation of a Source-Routing-Based SDN Data Plane," *IEEE Syst. J.*, vol. 15, no. 1, pp. 56–67, 2021.

[16] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel.Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, 1996.

[17] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[18] *OpenFlow Switch Specification, Version 1.3.3*, Open Networking Foundation, Sept. 2013.

[19] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *Proc. of the ACM SIGCOMM Workshop Hot Topics in Networks*, 2010.

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

[21] 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. of the IEEE INFOCOM Workshop*, 2017, pp. 559–564.

[22] J. M. Llopis, J. Pieczerak, and T. Janaszka, "Minimizing Latency of Critical Traffic through SDN," in *Proc. IEEE Int. Conf. Networking, Architecture and Storage*, 2016, pp. 1–6.

[23] H. Li, P. Li, and S. Guo, "MoRule: Optimized rule placement for mobile users in SDN-enabled access networks," in *Proc. of the IEEE GLOBECOM*, TX, Dec. 2014, pp. 4953–4958.