Sway: Traffic-Aware QoS Routing in Software-Defined IoT

Niloy Saha, *Student Member, IEEE,* Samaresh Bera, *Student Member, IEEE,* and Sudip Misra, *Senior Member, IEEE*

Abstract—In this paper, we propose a traffic-aware quality-of-service (QoS) routing scheme in software-defined internet of things (SDIoT) network. The proposed scheme exploits the unique features of software-defined networking (SDN), such as flow-based nature, and network flexibility, in order to fulfill QoS requirements of each flow in the network. We consider two types of QoS routing strategies — delay-sensitive and loss-sensitive — for incoming packets from end-devices in the network. The former is devised to deal with delay-sensitive flows, and the latter deals with loss-sensitive flows, in order to maximize the overall network performance. We propose a greedy approach based on Yen's K-shortest paths algorithm to compute the optimal forwarding path, while considering the QoS requirements of each packet. Consequently, the SDN controller deploys adequate flow-rules at the forwarding devices in the network. Extensive simulation results show that the proposed scheme significantly reduces the end-to-end delay and the percentage of flows which violate QoS constraints compared to the benchmarks considered in the study. It is also observed that the proposed scheme adequately satisfies the QoS requirements for both type of flows in contrast to the existing schemes. In particular, with 2000 flows in the network, the proposed scheme achieves 13%, 14% and 15% (with AttMpls topology) and 38%, 37% and 39% (with Goodnet topology) reduction in QoS violated flows as compared to the existing LARAC, SPD, and MRC schemes, respectively.

Index Terms—Software-Defined Networking, Internet of Things, Quality-of-Service, Routing

1 Introduction

The emerging internet of things (IoT) paradigm envisions a ubiquitous network of *smart* objects interconnected with one another using the Internet as a global platform [1]. Such a pervasive IoT network involves a massive number of heterogeneous smart objects, such as radio-frequency identification (RFID) tags, wireless sensors/actuators, and machine-to-machine (M2M) communication devices. The current Internet will play a key role in providing a global backbone for the interconnection of these objects across a varied range of communication technologies, capabilities, and requirements [1]. However, the massive influx of data [2] from billions of heterogeneous, interconnected *things*[1] and their varied application-dependent requirements will create greater demand on the underlying backbone network, and thus, require quality-of-service (QoS) which cannot be guaranteed by the current Internet designed for best-effort data transmission.

1.1 Motivation

The variety of IoT applications across multiple domains ranging from smart healthcare to vehicular automation, require diverse QoS guarantees from the network — in terms of throughput, delay, jitter, and reliability (packet-loss) [3]–[5]. Video-based applications for IoT are expected to increase seven times by 2021 [6]. These type of applications, particularly surveillance applications, can tolerate some amount of packet-loss but require timely information delivery. On the other hand, low-power and resource-constrained devices which are considered to be a major part of the IoT, are especially sensitive to packet-loss. Due to their constrained nature, these type of devices rely application-layer retransmissions to guarantee message delivery [7], [8]. Hence, they require enhanced support from the network in terms of packet-loss to minimize retransmissions and thereby conserve energy. Therefore, it is evident that applications may be *delay-sensitive* or *loss-sensitive* or *both*, depending on the requirements. The heterogeneous nature of IoT devices (in terms of memory, processing power, and energy) leads to varying QoS demands, even among the services requiring the same type of QoS guarantees. For example, latency requirements of factory automation can vary from 0.25 to 10 ms, whereas process automation can tolerate delay upto 100 ms [9]. Thus, the wide range of requirements suggest that there is a need to maintain application-dependent QoS guarantees in the network.

Software-defined networking (SDN) [10] is a promising approach to control the network in a unified manner and simplify network management by decoupling the data and control planes. A centralized controller abstracts away the control logic from the networking devices, which are converted into simple high-speed forwarding elements. The higher abstraction level and programmable APIs allow dynamic run-time reconfiguration of high level (application-level) QoS policies and routing, without being concerned about low-level hardware configuration. This makes SDN particularly attractive to address the problem of application-

• *N. Saha, S. Bera and S. Misra are with the Computer Science and Engineering Department, Indian Institute of Technology, Kharagpur, 721302, India, Email: niloysaha@iitkgp.ac.in, s.bera.1989@ieee.org, smisra@sit.iitkgp.ernet.in*

1. *Things* consist of devices such as radio-frequency identification (RFID) tags, wireless sensors and actuators, and mobile phones.

dependent QoS management.

Recent studies [4], [11]–[15] explored the advantages of SDN in QoS management. Some of them [11], [15] focused on a particular application such as video streaming. Others [13], [14] focused on data-center networks (DCNs) where only throughput and link-utilization were taken into account. The existing solutions mostly focused on a single metric (such as delay) or considered a linear combination of different QoS metrics, which fail to satisfy individual QoS requirements [16]. Further, they did not consider SDN rule-capacity and thus implicitly assumed that rule-capacity does not affect QoS.

In this paper, we propose a more general approach which considers the heterogeneous QoS requirements of IoT. In particular, our approach differs from the state-of-the-art in two aspects. First, we utilize the programmatic nature of SDN to route IoT traffic using application-specific requirements, i.e., whether they are delay-sensitive or loss-sensitive or both. Second, we take into account the effect of SDN rule-capacity on QoS routing.

1.2 Contribution

In this paper, we introduce a *traffic-aware* QoS routing scheme in Software-defined IoT (SDIoT) networks, named *Sway*, to address the above mentioned issues. The proposed scheme exploits the flow-based nature and the flexibility of SDN routing, in order to fulfill application-specific QoS requirements of flows such as either delay, loss, or both, as discussed in Section 1.1. We propose an efficient greedy heuristic based on Yen's K-shortest paths algorithm [17] to compute optimal routing paths, depending on QoS requirements of flows present in the network. Extensive simulation results are presented to show the effectiveness of the proposed scheme. In summary, the specific *contributions* in this work are as follows:

1) We propose a *traffic-aware* routing scheme to calculate QoS paths for incoming flows in an SDIoT network, while simultaneously taking into account the SDN rule-capacity constraint. The problem is challenging because of the varied QoS requirements of flows present in an IoT network.
2) We formulate the QoS routing problem in SDIoT network as an integer linear program (ILP). It takes into account the particular type of traffic and the associated QoS requirements, such as delay or packet-loss or both.
3) Since the ILP is NP-hard, we present an efficient greedy algorithm to solve the QoS routing problem in SDIoT networks.
4) We evaluate the proposed scheme using the POX SDN controller and the Mininet network emulator. In particular, with 2000 flows in the network, the proposed scheme achieves 13%, 14% and 15% (with AttMpls topology) and 38%, 37% and 39% (with Goodnet topology) reduction in QoS violated flows as compared to the existing LARAC, SPD, and MRC schemes, respectively.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we review the relevant state-of-the-art from the perspective of QoS routing in the SDIoT network. Section 3 describes the overall system architecture. In Section 4, we present the proposed approach towards traffic-aware routing in SDIoT network. In Section 5, simulation results are presented to show the effectiveness of the proposed scheme. We finally conclude the paper in Section 6, while citing directions for future work.

Table 1: Summary of existing literature on SDIoT

Work	Area of Focus	Remarks
Qin *et al.* [4] and Llopis *et al.* [12]	QoS routing for SDIoT considering only delay or throughput	There exists a research lacuna on satisfying the different QoS criteria in IoT such as either delay or loss or both while simultaneously addressing the rule-capacity constraint of SDN.
Gupta *et al.* [18], Tomovic *et al.* [19], and Muñoz *et al.* [20]	End-to-end orchestration and management of IoT services using SDN	
Hakiri *et al.* [21]	Traffic engineering in wireless fog routers using SDN	
Tang *et al.* [22]	Deep learning-based adaptive channel assignment for SDIoT	
Bellavista *et al.* [23]	Federated SDN controllers for IoT and FiWi access network	

2 Related Work

We discuss the state-of-the-art from three different perspectives — suitability of SDN in IoT, quality of service in IoT, and SDN-based QoS approaches.

2.1 Suitability of SDN in IoT

Recent studies [20], [22]–[27] have highlighted the advantages of a software-defined architecture for IoT — in terms of simplifying network management, adapting to meet the massive data onslaught and enabling diversified QoS. Tang *et al.* [22] proposed an adaptive channel assignment scheme for SDIoT in the presence of periodic and bursty IoT traffic. In the proposed approach, the centralized SDN controller was used to calculate the dynamic load using deep convolutional neural network (CNN). Subsequently, an adaptive channel assignment algorithm based on the load was used in order to reduce interference and improve the quality of transmission. Bellavista *et al.* [23] proposed a federated architecture for SDIoT, where the domain-specific SDN controllers for IoT network and fiber-wireless (FiWi) access network were capable of exchanging monitoring and control data for better end-to-end QoS management.

Concurrently, fog computing was introduced to meet the challenges of IoT such as increasing demands of real-time delay-sensitive applications, by moving the processing closer to the edge. More recently, due to the advantages of SDN in making the network flexible and programmable, it is being applied for orchestration and management of fog-based IoT architectures [18], [19], [21]. Tomovic *et al.* [19] proposed a fog-based architecture for IoT consisting of hierarchical geo-distributed fog nodes. Hakiri *et al.* [21] presented a SDN-based architecture for traffic engineering

in wireless fog networks. Muñoz *et al.* [20] presented an IoT-aware SDN and cloud orchestration architecture to distribute IoT analytics between the core and the edge. The authors proposed a hierarchical SDN controller to detect link congestion and notify the global service orchestrator which triggers provision of distributed IoT analytics at the edge/fog nodes. We argue that the proposed solution is complementary to the fog-based IoT architectures. Further, with fog computing leveraged to handle requests from delay-sensitive applications, due to network load on the fog nodes, it may be required to reroute delay-sensitive requests among the fog devices. Therefore, it is required to adequately select the minimum latency path to serve the delay-sensitive requests based on dynamic network conditions. In this work, we propose a traffic-aware QoS routing scheme to forward traffic coming from IoT devices at the backbone network, while considering the requirements. It is noteworthy that the proposed scheme is generalized from the perspective of IoT network enabled with SDN. Hence, the proposed QoS routing scheme in SDIoT networks is useful in conjunction with fog-based architectures for IoT to further enhance the QoS.

2.2 Quality of Service in IoT

Duan et al. [28] considered a layered IoT architecture and analyzed the QoS requirements at different layers — perception, network, and service and management layers. Jin et al. [3] proposed four architectural approaches to an IoT network for smart city applications and discussed the required network QoS. The authors affirmed that delay and packet-loss are important QoS metrics that must be considered at the IoT network layer. The discussions in [3], [28] mainly focused on high-level architectural issues QoS requirement analysis. However, details on how QoS may be achieved was not discussed. Awan et al. [29] proposed a finite-capacity queuing system to provide preferential treatment to delay-sensitive IoT traffic originating from resource-constrained devices. The authors focused on a packet-level queuing-based approach which is complementary to the proposed scheme. Particularly, we consider a flow-level QoS routing scheme instead of packet-level.

2.3 SDN-based QoS Routing

Egilmez et al. [11] considered a per-flow QoS routing approach for scalable video-streaming applications in SDN. The authors considered video streams consisting of one base layer and one or more enhancement layers. The authors utilized the per-flow routing capabilities of SDN to provide prioritized forwarding of the base layer video in order to have continuous video playback at the receiver. However, they considered delay-variation (jitter) as the only routing constraint and did not take into account the rule-capacity constraints of OpenFlow switches. Additionally, they considered a linear combination of an *additive* metric (delay) and a *multiplicative* metric (loss), which may lead to non-polynomial complexity and failure to satisfy individual QoS constraints [16].

Qin *et al.* [4] presented a genetic algorithm-based flow scheduling approach for software-defined IoT. The authors designed a fitness function based on the delay, jitter and bandwidth requirements to iteratively calculate QoS path for IoT flows. Llopis *et al.* [12] presented a software-defined approach to minimize latency of critical traffic in IoT. The authors presented realistic results using SDN controller and network emulator to show the feasibility of their scheme. However, key factors such as packet-loss and rule-capacity were not considered.

Table 2: Summary of key notations

Notation	Description
$\mathcal{S}$	Set of OpenFlow-enabled switches.
$\mathcal{L}$	Set of links between the switches.
$\mathcal{F}$	Set of all flows.
$\mathcal{N}(i)$	Neighbor set of switch $i \in \mathcal{S}$.
$x_k(i,j)$	Indicator function to denote whether flow f_k is routed on link $(i,j) \in \mathcal{L}$.
$R(i)$	Number of rules at switch $i \in \mathcal{S}$.
$D(f_k)$	Cumulative delay experienced by flow f_k.
$L(f_k)$	Cumulative loss experienced by flow f_k.
$C(f_k)$	Capacity of flow f_k.
$C_{\text{res}}(i,j)$	Residual capacity of link $(i,j) \in \mathcal{L}$.
$R_{\max}$	Max. rule-capacity of an OpenFlow switch.

Cohen et al. [13] and Huang et al. [14] considered traffic engineering for SDN using multipath routing, while considering TCAM rule-capacity constraints. As the problem is NP-hard, the authors proposed efficient randomized approximation algorithms and heuristic schemes to find multiple paths while avoiding rule duplication at switches lying on common paths. However, the authors considered throughput as the only QoS metric of concern, which is not sufficient for IoT. Liu et al. [15] considered delay-optimized multipath routing for video streaming applications in software-defined inter-DCNs. They proposed a scheme to yield *sparse* solutions, which limits the number of paths over which traffic is split, in order to minimize rule-duplication at the switches.

In contrast, we consider single-path routing as IoT traffic is usually low-rate [30]. Consequently, in our work, rule-capacity constraints result from a large number of IoT flows, rather than splitting flows over multiple paths. Additionally, we consider delay and packet-loss as QoS metrics along with energy and rule-capacity constraints.

In Table 1, we summarize the existing literature on SDIoT. Detailed *synthesis* of the existing literature reveals that there exists a research lacuna on exploiting the flexibility of SDN to provide QoS based on network flow-type in IoT, while simultaneously considering the additional constraints imposed by SDN-based architecture. Consequently, we propose a traffic-aware QoS routing scheme in SDIoT.

3 System Model

In this Section, we present the system model of software-defined IoT network considered in this work. The notations used in this work are summarized in Table 2.

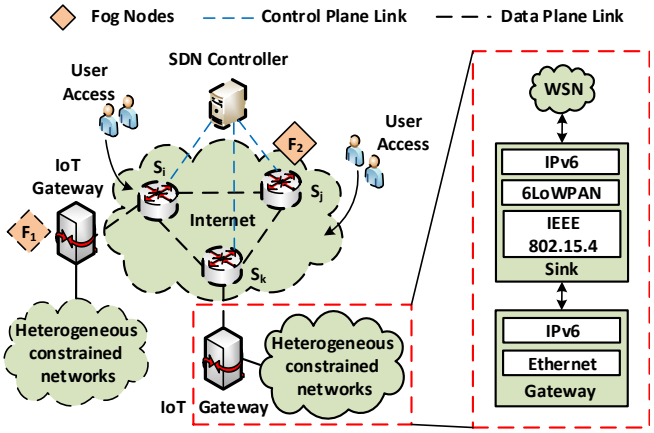


Figure 1: SDN-based system architecture

3.1 Architecture

We consider a ubiquitous connectivity model for IoT as discussed in [3]. In this model, heterogeneous constrained networks [31] are connected to the Internet over the same IP-enabled Internet backbone, through the use of IoT gateways [32][2]. To address gateway to Internet connectivity, hybrid communication architectures combining low-latency fiber-optic and heterogeneous wireless technology may be used [33], [34]. The IoT gateway and the SDN-enabled switches operate as fog nodes [19] to bring processing closer to the edge and reduce load on the core network. Figure 1 shows the SDIoT architecture considered in this work.

Let us consider an SDN-enabled network as a directed graph $\mathcal{G} = (\mathcal{S}, \mathcal{L})$, where $\mathcal{S}$ denotes the set of all SDN-enabled switches, and $\mathcal{L} = \{(i,j) \mid (i,j) \subset \mathcal{S} \times \mathcal{S}, i \neq j\}$ denotes the set of links between them. The SDN controller communicates with the switches through the OpenFlow protocol [10], and communication between the controller and application layer is achieved through SDN northbound application programming interface (API).

3.2 Heterogeneous Traffic and its Impact on QoS

Typically, IoT applications are event-driven (such as smart alarm system) or low-rate (such as health-monitoring), and have low bandwidth requirements. However, due to their importance, they generally have stringent QoS requirements [35]. IoT applications may use either TCP or UDP as the underlying transport layer protocol. However, due to complexity of running TCP on constrained devices, IoT protocols such as CoAP [8] and MQTT-SN [7] have been developed which use UDP as transport. The use of UDP is also suitable for delay-sensitive applications as it eliminates additional latency due to connection setup and re-transmission. Applications can implement their own re-transmission scheme[3] depending on the requirements.

In the ubiquitous IoT model, constrained devices are connected to application servers, through IoT gateways, and over long range access-core networks. Consequently, traffic from IoT applications compete with traditional Internet traffic over the same shared network. Currently, majority of the Internet traffic is carried by TCP-based protocols like HTTP and FTP. Competing with aggressive TCP flows (such as bulk transfers or video streaming) can significantly affect the QoS of these low-rate UDP-based IoT applications. In contrast to high-rate UDP flows, which prevent TCP flows from increasing the congestion window size, low-rate UDP flows experience increased packet-loss rates in the presence of greedy TCP flows [36].

Increasingly, there are use-cases of latency sensitive IoT applications [9], whose QoS needs cannot be adequately met using IoT protocols like CoAP and MQTT, since they provide no guarantee over end-to-end latency. Application layer re-transmission in case of packet-loss may not be sufficient to meet the stringent latency requirements in these cases. Further, traffic flows from each such use-case has associated latency requirements which vary significantly, depending on the application. Thus, these is a need to choose an adequate end-to-end path in terms of latency for each individual flow. We broadly classify such traffic as *delay-sensitive (ds)* flows. For example, video-streaming from surveillance applications, traffic from real-time health monitoring systems and 'smart' connected cars can be considered can be considered as delay-sensitive flows. While the influx of delay-sensitive traffic in IoT may not be massive, nevertheless, such traffic may originate from critical processes such as robotic arm in tele-surgery and road monitoring where meeting the latency constraints is of great importance [12]. However, benefiting such delay-sensitive flows too much may lead to severe degradation of QoS for other flows.

On the other hand, the majority of IoT applications are delay-tolerant but may benefit from increased reliability in terms of packet loss. Since such applications involve energy-constrained devices, reliability provided by application layer re-transmissions may prove costly in terms of energy efficiency. For example, wireless sensor networks (WSNs) used as part of an environmental monitoring system or wireless body area networks (WBANs) used as part of e-health system consist of energy-constrained devices. As these applications can tolerate some amount of delay, the focus should be on choosing the best end-to-end path in terms of packet-loss in order to minimize the number of application layer re-transmissions, while using an unreliable transport such as UDP. We broadly classify such traffic as *loss-sensitive (ls)* flows.

Mathematically, the set of flows $\mathcal{F}$ is given as :

$$\mathcal{F} = \{f_k \mid k \in \mathbb{N}\}, \quad f_k := (s_k, t_k, \lambda_k, q_k) \tag{1}$$

where s_k , t_k, q_k and λ_k denote the source, destination, QoS demand and the *type* (either *ds* or *ls*) of the k^{th} flow, f_k.

3.3 Routing Metric Selection

The composition rules of different metrics significantly affect the complexity of QoS routing [16]. Consequently, we evaluate the suitability of single *composite-metric* and multiple *single-metric* based schemes for QoS routing in

2. To this extent, the IETF has proposed the 6LoWPAN standard which enables IPv6 over IEEE 802.15.4 networks for constrained devices.

3. For example, MQTT has three levels of QoS defined as — delivered at most once, at least once, and exactly once.

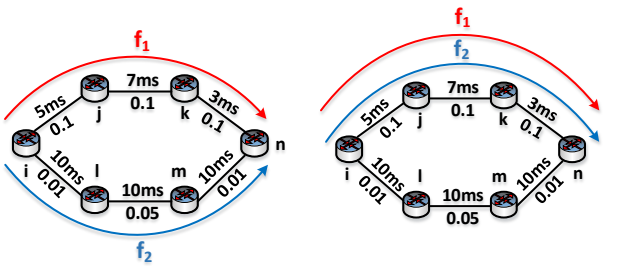


Figure 2: Illustrative example of the proposed scheme — (a) single-metric (delay-based) routing scheme; and (b) the proposed routing scheme

software-defined IoT. For the given network, the QoS metrics associated with each link $(i,j) \in \mathcal{L}$ are delay, $d(i,j)$, packet-loss probability, $l(i,j)$, and available bandwidth, $c(i,j)$. Delay is an additive metric, while bandwidth is concave. The composition rule for packet-loss probability is more complex, and hence, we consider the logarithm of success-probability (which itself is multiplicative), denoted by $\hat{l}(i,j)$, where $\hat{l}(i,j) = \log(1-l(i,j))$. Hence, for any path $\mathcal{P} = \{i,j,k,\cdots,s,t\}$, we have

$$D_{\mathcal{P}} = d(i,j) + d(j,k) + \cdots + d(s,t) \tag{2a}$$
$$C_{\mathcal{P}} = \min\{c(i,j), c(j,k) \cdots, c(s,t)\} \tag{2b}$$
$$\hat{L}_{\mathcal{P}} = \hat{l}(i,j) + \hat{l}(j,k) + \cdots + \hat{l}(s,t) \tag{2c}$$

The cumulative packet-loss probability is represented mathematically, as $L_{\mathcal{P}} = 1-\exp(\hat{L}_{\mathcal{P}})$. It is intuitive that consideration of multiple metrics for QoS routing gives an accurate model of the network, and therefore, offers better performance compared to single-metric based schemes. However, finding multi-constrained QoS routing paths based on metrics with different composition rules is NP-hard [16]. We utilize the dynamic run-time reconfiguration capabilities of SDN, to run separate *single-metric* based, polynomial-time QoS routing algorithms in parallel, based on the particular QoS requirements of the *type* of traffic. This, in effect, leads to an accurate model of the QoS requirements of the network, while reducing the complexity of QoS routing.

Example 1. *We demonstrate the proposed routing scheme with an example depicted in Figure 2. We consider two paths, $\mathcal{P}_1 = \{i,j,k,n\}$ and $\mathcal{P}_2 = \{i,l,m,n\}$. The cumulative delay and packet-loss probability associated with each path are $D_{\mathcal{P}_1} = 15ms$, $D_{\mathcal{P}_2} = 30ms$ and $L_{\mathcal{P}_1} = 0.271$, $L_{\mathcal{P}_2} = 0.069$. Let us consider two flows, $f_1 = (s_1 \leftarrow i, t_1 \leftarrow n, \lambda_1 \leftarrow ds, q_k^{delay} \leftarrow 20ms)$ and $f_2 = (s_2 \leftarrow i, t_2 \leftarrow n, \lambda_2 \leftarrow ls, q_k^{loss} \leftarrow 0.1)$. The delay-based scheme (refer Figure 2(a) does not take packet-loss criterion of f_2 into account, and hence, $\mathcal{P}_1$ with path-loss $L_{\mathcal{P}_1} = 0.271$ fails to satisfy QoS requirements of f_2, which is, therefore, forwarded using best-effort. On the other hand, the proposed scheme (refer Figure 2(b)) selects two different paths, $\mathcal{P}_1$ and $\mathcal{P}_2$, for f_1 and f_2, respectively, depending upon their individual QoS requirements and link characteristics.*

From the example, it is evident how SDN may be utilized to provide optimal QoS routing paths depending on the type of traffic, while maintaining low complexity. However, OpenFlow [10], the *de-facto* standard used to realize such an SDN architecture, introduces a new challenge — in terms of number of forwarding rules that can be placed at a switch. In the subsequent section, we discuss the impact of limited rule-capacity on SDN based QoS routing.

3.4 Rule-Capacity Constraint

OpenFlow switches employ ternary content addressable memory (TCAM), which enables fast lookup of multiple fields in a parallel fashion. However, TCAM has several drawbacks, including high power consumption and increased cost. Therefore, the total number of rules that can be inserted at a particular OpenFlow switch is constrained by the available TCAM memory [37]. This constraint[4] affects the energy awareness of the proposed scheme, as shown in Figure 3. To minimize the number of active links in the network, the activated links should be preferred over the inactive ones, while considering the QoS requirements, as shown in Figure 3(a). However, rule-capacity overflow leads to the activation of inactive links, and thereby increases the cost, as shown in Figures 3(b) and 3(c).

In Section 4, we discuss traffic-aware routing optimization to route multiple flows in an efficient manner, while taking into consideration the effect of multiple traffic types and the rule-capacity constraints of an SDIoT network.

4 Traffic-Aware Routing Optimization

In this Section, we formulate the routing problem as an integer linear program (ILP) and a greedy heuristic algorithm is proposed to efficiently solve the QoS routing problem for the given SDIoT network. The objective is to find a set of links, $\{(i,j) \in \mathcal{L}\}$, on which *ds* and *ls* flows are routed, in order to minimize the associated cost, while considering allowable delay and packet-loss. Additionally, rule-capacity constraint at the switches must be satisfied. The prerequisites required to formulate the ILP are developed in Section 4.1.

4.1 Prerequisites

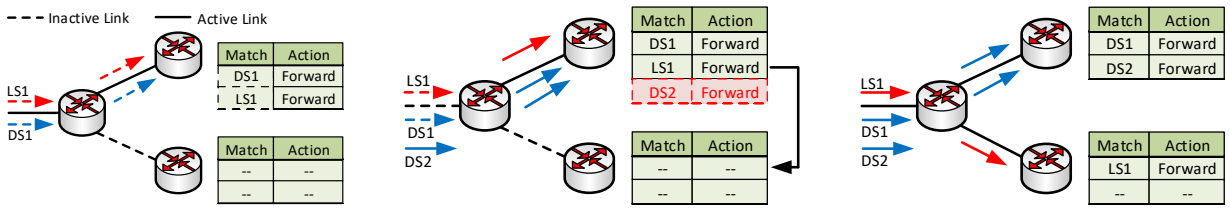
To represent whether a particular flow is routed on a specific link, $(i,j) \in \mathcal{L}$, we define an identity function, $x_k(i,j)$, such that

$$x_k(i,j) := \begin{cases} 1, & \text{if } f_k \text{ is routed on } (i,j) \in \mathcal{L} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The neighborhood of a switch $i \in \mathcal{S}$ is denoted by $\mathcal{N}(i)$, which represents the set of all switches that are adjacent (connected directly) to it. A flow f_k is routed on link $(i,j) \in \mathcal{L}$, only if there exists a distinct rule[5] at switch $i \in \mathcal{S}$ to any switch $j \in \mathcal{N}(i)$, $i \neq j$. Therefore, the number of rules present at switch $i \in \mathcal{S}$ associated with

4. Since our objective is QoS routing of IoT flows, we limit our discussion on rule-capacity constraints of TCAM. Interested readers may refer to [37] for details.

5. We consider an exact-match rule placement strategy where a particular rule is defined by ⟨ in-port, src-mac, dst-mac, src-ip, dst-ip, src-port, dst-port ⟩



(a) Flows routed through active link (b) Rule-capacity overflow (c) Activation of inactive link

Figure 3: Effect of rule-capacity constraint on routing of flows

flow f_k is given as $R_k(i) = \sum_{j \in \mathcal{N}(i)} x_k(i,j)$. Consequently, the number of rules present at a particular switch $i \in \mathcal{S}$ is given mathematically as:

$$R(i) = \sum_{f_k \in \mathcal{F}} R_k(i) \tag{4}$$

From the composition rules of delay and packet-loss, the total delay and total success probability of a flow f_k are given as :

$$D(f_k) = \sum_{(i,j) \in \mathcal{L}} d(i,j) x_k(i,j) \tag{5a}$$

$$\hat{L}(f_k) = \sum_{(i,j) \in \mathcal{L}} \hat{l}(i,j) x_k(i,j) \tag{5b}$$

From the bandwidth composition rule, the capacity of a path, over which a flow is routed, is defined mathematically, as $C(f_k) = \min_{(i,j) \in \mathcal{L}} \{c(i,j) x_k(i,j)\}$

The residual bandwidth of a link, $(i,j) \in \mathcal{L}$, denoted by $C_{\text{res}}(i,j)$, is the bandwidth remaining after all flows $f_k \in \mathcal{F}$ with their associated bandwidth requirements, are routed through the link, $(i,j) \in \mathcal{L}$. Mathematically,

$$C_{\text{res}}(i,j) = c(i,j) - \sum_{f_k \in \mathcal{F}} q_k^{\text{bandwidth}} x_k(i,j) \tag{6}$$

4.2 Optimization Model

In this Section, we formulate the QoS routing problem as an integer linear program (ILP), with the help of the equations presented in Section 4.1. We define two separate cost functions, $f_d(\cdot)$ and $f_{\hat{l}}(\cdot)$, for *ds* and *ls* flows, respectively, as follows:

$$f_d(k) := \sum_{(i,j) \in \mathcal{L}} \Big(d(i,j) x_k(i,j) + \alpha x_k(i,j) \Big) \tag{7a}$$

$$f_{\hat{l}}(k) := \sum_{(i,j) \in \mathcal{L}} \Big(\hat{l}(i,j) x_k(i,j) - \beta x_k(i,j) \Big) \tag{7b}$$

Equations (7a) denotes the cost factor for routing *ds* flows, and comprises of the cumulative delay experienced by flow f_k along with a penalty factor. The penalty factor $\sum_{(i,j) \in \mathcal{L}} x_k(i,j)$ denotes the sum of activated links for flow f_k and is introduced to prescribe high cost to $f_d(k)$ and $f_{\hat{l}}(k)$ if large number of links are activated. The objective of the penalty factor is to route as many *ds* and *ls* flows as possible through the existing activated links in order to achieve optimal energy consumption. As long as the QoS constraints are satisfied, the cost factor $f_d(k)$ aims to minimize the cumulative delay of the chosen subset of links. Similarly, we formulate Equation (7b) with an aim to minimize the cumulative packet-loss over the chosen links. It is noteworthy that in accordance with the loss-composition rule stated in Equation (2c), we use the logarithm of success-probability $\hat{l}(i,j)$, in Equation (7b). Therefore, equivalently, the cost function in Equation (7b) aims to maximize the cumulative success-probability. In Equations (7a) and (7b), α and β denote normalizing constants.

The objective of the SDN-controller is to find the optimal set of links, while simultaneously minimizing the energy consumption. Consequently, we formulate the choice of appropriate links, on which to route the heterogeneous flows, taking into account their type and QoS requirements, as an optimization problem, as follows:

$$\min_{f_k} \; c_1 \sum_{f_k \in \mathcal{F} | \lambda_k = ds} f_d(k) - c_2 \sum_{f_k \in \mathcal{F} | \lambda_k = ls} f_{\hat{l}}(k) \tag{8a}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \mathcal{L}} x_k(i,j) - \sum_{(j,i) \in \mathcal{L}} x_k(j,i) = \begin{cases} +1, & \text{if } i = s_k, \\ -1, & \text{if } i = t_k, \\ 0, & \text{if } i \neq s_k, t_k, \end{cases}$$

$$\forall i \in \mathcal{S}, f_k \in \mathcal{F}, \tag{8b}$$

$$D(f_k) \leq q_k^{\text{delay}}, \quad \forall f_k \in \mathcal{F}, \tag{8c}$$

$$L(f_k) \leq q_k^{\text{loss}}, \quad \forall f_k \in \mathcal{F}, \tag{8d}$$

$$R(i) \leq R_{\max}, \quad \forall i \in \mathcal{S}, \tag{8e}$$

$$C(f_k) \leq q_k^{\text{bandwidth}}, \quad \forall f_k \in \mathcal{F}, \tag{8f}$$

$$C_{\text{res}}(i,j) \geq 0 \quad \forall (i,j) \in \mathcal{L} \tag{8g}$$

Equation (8a) is the *objective function* to be minimized, where c_1 and c_2 represent constants depending on the application. By varying c_1 and c_2 it is possible to prioritize either *ds* or *ls* flows. Equation (8b) represents the *conservation of flow constraints* which specify that every flow $f_k \in \mathcal{F}$ has only one source and sink node. Equations (8c) and (8d) represent the *delay constraints* and *loss constraints*, where q_k^{delay} and q_k^{loss} represent the QoS demands of the flow f_k in terms of delay and loss. It is noteworthy that Equations (8c) and (8d) take into account the packet-loss requirements of delay-sensitive flows and vice-versa. The delay and loss are specified using the northbound API of the SDN controller according to the application-specific requirements [38]. Equation (8e) represents the *rule-capacity constraints* where $R_{\max}$ is the maximum number of rules that can be installed at any OpenFlow switch $i \in \mathcal{S}$. Equation (8f) represents the *demand constraints* where $q_k^{\text{bandwidth}}$ denotes the QoS demand of f_k in terms of bandwidth. Finally, Equation (8g) represents the

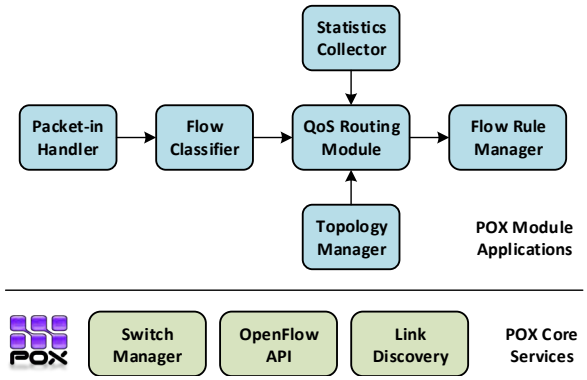


Figure 4: Proposed controller architecture

capacity constraints associated with each link $(i, j) \in \mathcal{L}$. If the residual capacity of a link is negative, then the link is incapable of supporting the given routing configuration. Therefore, non-negativity is imposed on residual capacity associated with a link, using the *capacity constraints*.

4.3 Algorithm for Traffic-Aware Routing

Since the ILP formulated in Section is NP-hard in general, we propose an efficient greedy heuristic based on Yen's K-shortest path algorithm [17] to compute the optimal routing path in order to maximize the overall network performance.

Figure 4 presents the proposed architecture of the SDN controller. The *Packet-in Handler* module is used to capture OpenFlow packet-in messages from the switches. The *Flow Classifier* module is used to classify traffic according to their type. We adopt the existing flow classification scheme proposed by Ng et al. [39]. The *Statistics Collector* module is used to collect statistics about the flows such as delay and packet-loss. We use active probing to determine the link delay; the SDN controller injects probe packets at the source switch $i \in \mathcal{S}$, and after receiving them, the destination switch $j \in \mathcal{S}$, sends them back to the controller. The controller determines the delay from the difference in arrival and departure times of the probe packet, while taking into account the switch-controller latency [40]. Packet loss is determined by the difference in OpenFlow port statistics of the source and destination switches. [40][6]. The *Topology Manager* module is used to maintain a global view of the network which is input to the *QoS Routing Module*. The *QoS Routing Module* implements Algorithm 1 and computes optimal routing paths. After path calculation, the *Flow Rule Manager* module is used to place appropriate flow-rules along all the switches lying on the path. The *Flow Rule Manager* is also used to update the rule-capacity of the switches.

Assumption 1. *Queuing and processing delays are considered to be the same for all switches.*

6. We limit our discussion on calculating the delay and packet-loss in a link, as our objective is to route the flows according to their QoS requirements. Interested readers may refer to the scheme proposed by Adrichem et al. [40] for details.

Assumption 2. *We assume a Bernoulli model for the link loss, where each link drops packets with some fixed probability, independent of the other links. We consider the packet loss probability to be uniformly distributed across all links, within a given range.*

Assumption 3. *We consider the OpenFlow switches in the network to be homogeneous, i.e., rule-space across all the switches present in the network is assumed to be uniform.*

Algorithm 1 Routing algorithm for *ds* and *ls* flows

Inputs:
Graph, $\mathcal{G}$
Set of flows $\mathcal{F}$ with their associated QoS requirements
Maximum rules at a switch $R_{\max}$
Priority of *ds* flows, c_1, and *ls* flows, c_2 ▷ User defined
Output:
Set of Paths $\{\mathcal{P}\}$ on which *ds* flows and *ls* flows can be routed

1: **for** each $j \in \mathcal{G}.\mathcal{S}$ **do**
2: $rules(j) \leftarrow R_{\max}$ ▷ Initialize rule-capacity
3: m, m $\leftarrow 1$
4: **while** all flows $f_k \in \mathcal{F}$ have not been routed **do**
▷ Alternate *ds* and *ls* flows to ensure fair allocation
5: **if** $\exists$ *ds* flows not routed **then**
6: **for** $i \leftarrow 1$ to c_1 **do**
7: path $\leftarrow$ GET-QOS-PATH$(s_m, t_m, \lambda_m \leftarrow ds, q_m)$
8: m $\leftarrow$ m + 1 ▷ Route m^{th} *ds* flow using path
9: **if** $\exists$ *ls* flows not routed **then**
10: **for** $j \leftarrow 1$ to c_2 **do**
11: path $\leftarrow$ GET-QOS-PATH$(s_n, t_n, \lambda_n \leftarrow ls, q_n)$
12: n $\leftarrow$ n + 1 ▷ Route n^{th} *ls* flow using path
13: **function** GET-QOS-PATH(s, t, λ, q)
14: **if** type = *ds* **then**
15: **for** path in K-SHORTEST-PATHS(s, t, f_d) **do**
16: **if** CHECK-QOS(path, q) **then** ▷ QoS satisfied
17: best-path $\leftarrow$ path
18: **else if** type = *ls* **then**
19: **for** path in K-SHORTEST-PATHS(s, t, f_l) **do**
20: **if** CHECK-QOS(path, q) **then** ▷ QoS satisfied
21: best-path $\leftarrow$ path
return best-path
22: **function** CHECK-QOS(path, demand)
23: **if** CHECK-DELAY **and** CHECK-LOSS **and** CHECK-BANDWIDTH **and** CHECK-RULES **then**
24: **return** True
25: **return** False

Algorithm 1 takes the cost of each link in terms of delay or packet loss as inputs and returns a feasible routing (if exists) for the set of *ds* and *ls* flows. The *ds* and *ls* flows are served according to the priority associated with each type, and are application-dependent. This round-robin type serving scheme, which is dependent on user-defined priorities, ensures fairness among the traffic types. In steps 8 and 12, the *ds* and *ls* flows are routed according to the paths calculated. The available rule-capacity of each switch and residual bandwidth of each link are updated in accordance with Equations (4) and (6), respectively. The function GET-QOS-PATH incorporates Yen's K-Shortest path algorithm [17] to return k shortest paths ranked according to the cost functions for *ds* and *ls* flows defined in Equations (7a) and (7b). Steps 15 and 19 along with function CHECK-QOS are used to take into consideration the loss-constraints of *ds* flows and vice-versa.

Table 3: Simulation parameters

Parameter	Value
Topology	AttMpls , Goodnet [41]
Flow bandwidth	0.20 - 0.40 kbps [42]
Number of switches	25 (AttMpls), 17 (Goodnet) [41]
Number of links	57 (AttMpls), 31 (Goodnet) [41]
Max. delay *ds* flow	0.25–100 ms [9]
Max. delay *ls* flow	1.6–10 s [43]
Max. loss *ds* flow	10^{-5}–10^{-3} [9]
Max. loss *ls* flow	10^{-9}–10^{-8} [9]
Ratio ds & ls flows	1:10 to 1:50
Avg. packet size	94 – 699 bytes [30]
Active volume	142 – 27,716 bytes [30]
Mean rate	562 - 516,540 bps [30]
Active time	1 – 34 s [30]

We analyze the complexity of the proposed scheme by considering the time complexity of Algorithm 1. The while loop in step 4 runs $|\mathcal{F}|$ times, and in each iteration, it makes a call to GET-QOS-PATH. Since the shortest-path computations from Yen's algorithm are the most expensive operations in terms of time complexity, the running time of GET-QOS-PATH can be upper-bounded by $\mathcal{O}(\mathrm{K}|\mathcal{S}|(|\mathcal{L}| + |\mathcal{S}|\log|\mathcal{S}|))$ (assuming an adjacency-list representation of the network graph $\mathcal{G}$). Therefore, using aggregate analysis, the time-complexity of the proposed scheme is $\mathcal{O}((|\mathcal{F}|) * (\mathrm{K}|\mathcal{S}|(|\mathcal{L}| + |\mathcal{S}|\log|\mathcal{S}|)))$. Hence, the proposed scheme runs in pseudo-polynomial time.

5 Performance Evaluation

5.1 Simulation Settings

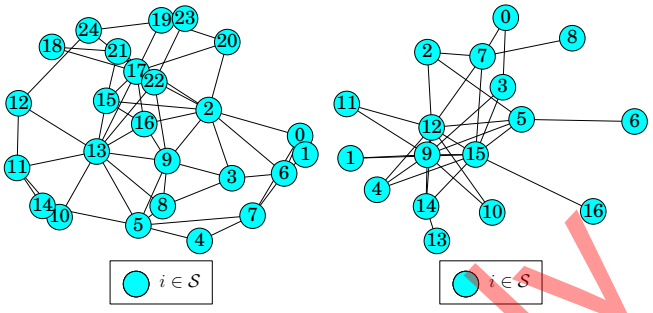
We evaluate the proposed scheme using the POX SDN controller[7] and the Mininet network emulator [44]. All the experiments were carried out in a PC with Intel® CoreTM i7 CPU, 2.67 GHz processors and 8GB RAM running Linux kernel 4.4.0-103-generic. Different simulation parameters are considered as mentioned in Table 3. For our experiment, we consider two existing network topologies — AttMpls and Goodnet — from the Internet Topology Zoo [41]. We used the D-ITG traffic generator [45] to model IoT traffic flows from real traces presented in [30].

5.2 Benchmark Schemes

We compare the proposed scheme in terms of the performance metrics defined in Section 5.3 with baselines: Lagrangian Relaxation Based Aggregated Cost (LARAC) proposed in [11], Shortest Path Delay (SPD) proposed in [12], and Minimum Occupied Rule Capacity (MRC). The LARAC scheme uses a genetic algorithm (GA)-based approach to iteratively calculate the best QoS path. The SPD scheme considers the cost metric to be a function of link delay, where at every iteration, the next hop link with minimal delay is chosen. The MRC algorithm is based on a greedy heuristic, where at every switch, the neighboring switch with the maximum residual rule-capacity is chosen to be included in the routing path. This round-robin nature of MRC algorithm aims to evenly distribute the available rule-capacity of the network.

On the other hand, the proposed scheme[8], *Sway*, considers the application-specific QoS requirements for each flow and calculates the K-shortest paths according to the type of flow. It then iterates though the K paths sorted on the basis of type i.e. delay or loss till the QoS requirements are met.

7. https://github.com/noxrepo/



(a) AttMpls topology (b) Goodnet topology

Figure 5: Topologies considered for experiment

5.3 Performance Metrics

We consider the following performance metrics to evaluate the proposed scheme with the existing SPF-based and MRC schemes.

(i) *End-to-end delay*: We evaluate the average end-to-end delay experienced by each flow in the network given as $\sum_{f_k \in \mathcal{F}} D(f_k)/|\mathcal{F}|$.

(ii) *QoS violated flows*: We evaluate the QoS violated flows [9] in the network given as $(|\mathcal{F}| - p - q)$ where $p \subset \{f_k \mid \lambda_k = ds\}$ and $q \subset \{f_k \mid \lambda_k = ls\}$ denote the number of feasible *ds* and *ls* flows, respectively.

(iii) *Activated links*: We evaluate the active links in the network given as $\sum_{(i,j)\in\mathcal{L}} \left(1 - \prod_{f_k \in \mathcal{F}} \left(1 - x_k(i,j)\right)\right)$.

5.4 Results and Discussion

5.4.1 Sway vs ILP

We used the Gurobi Optimizer [46] to solve ILP formulated in Section 4.2. Figure 6 shows the comparison between the proposed scheme, *Sway*, and the ILP-based solution. We observe that the proposed greedy approach takes less time compared to the ILP while having moderate deviation from the optimal solution. Thus, the proposed greedy approach offers adequate tradeoff between runtime and consideration of QoS criteria.

5.4.2 End-to-end Delay

We analyze the end-to-end delay experienced by flows in the network. Figure 7 shows the performance of *Sway* compared to the benchmark schemes. From the figure, it is evident that even with increasing number of flows, the proposed scheme outperforms the benchmark schemes in

8. The proposed scheme and the term *Sway* are used interchangeably.

9. Flows having a routing path which do not satisfy the any one of delay, packet-loss, bandwidth constraints and rule-capacity constraints.

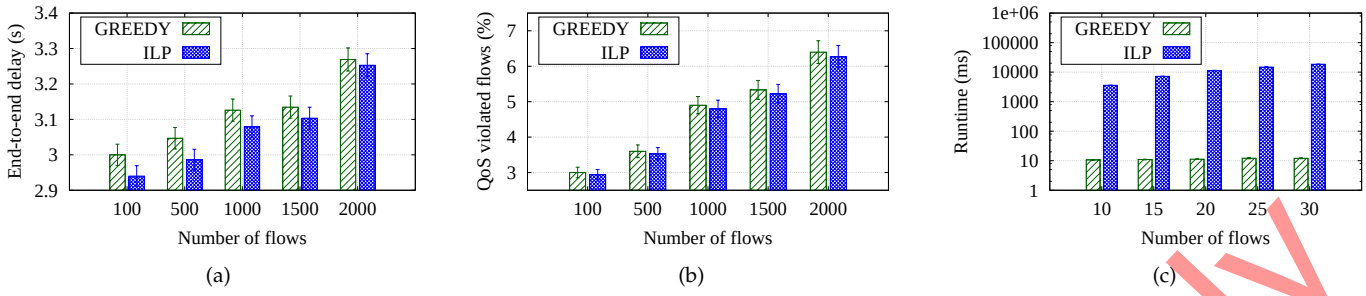


Figure 6: Comparison of *Sway* vs ILP-based solution — (a) end-to-end delay; (b) percentage of dropped flows; and (c) run-time

terms of end-to-end delay. In particular, *Sway* achieves 9%, 6%, and 10% (with AttMpls topology) and 11%, 4%, and 8% (with Goodnet topology) reduction in end-to-end delay (on average) as compared to LARAC, SPD and MRC schemes, respectively. This is indirectly dependent on the number of QoS violated flows, as they are routed using best-effort routing. Hence, a smaller percentage of QoS violated flows implies less end-to-end delay. The proposed scheme achieves optimal routing by alternating between the *ds* and *ls* flows. By routing *ds* flows using delay as routing metric and *ls* flows using loss as routing metric, it can simultaneously satisfy the QoS requirements of maximum number of flows.

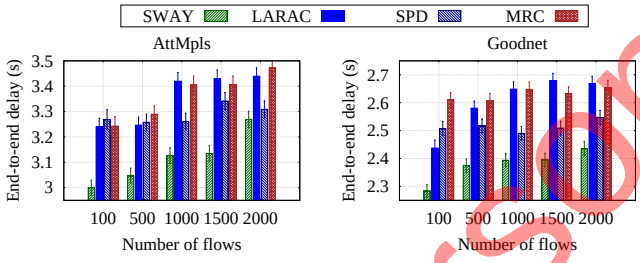


Figure 7: End-to-end delay in the network with varying number of flows

Figure 8 shows the end-to-end delay with varying ratio of *ds* and *ls* flows. It is evident that while the performance of the proposed scheme varies with different ratio of *ds* and *ls* flows, it outperforms the benchmark schemes (LARAC, SPD, and MRC) in all the cases.

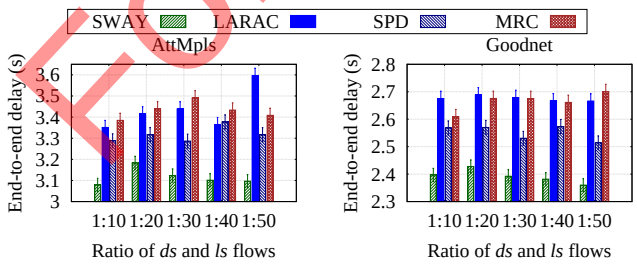


Figure 8: End-to-end delay in the network with varying ratio of *ds* and *ls* flows (1500 flows)

5.4.3 QoS Violated Flows

We analyze the percentage of QoS violated flows which have no feasible routing path and are therefore routed using best-effort. Figure 9 shows the performance of *Sway* compared to the benchmark schemes. We observe that the proposed scheme outperforms the existing schemes in terms of QoS violated flows as well. It is noteworthy that with an increase in the number of flows in the network, the proposed scheme significantly outperforms the benchmark schemes in terms of QoS violated flows. In particular, with 2000 flows in the network, *Sway* achieves 13%, 14% and 15% (with AttMpls topology) and 38%, 37% and 39% (with Goodnet topology) reduction in QoS violated flows as compared to the LARAC, SPD and MRC benchmark schemes.

In particular, from Figure 9, we observe that the LARAC, SPD and MRC schemes have a spike in QoS violated flows with 2000 flows in the network. This is due to the fact that the rule-capacity of the SDN switches is exceeded, resulting in flows passing through the switches to be dropped. Moreover, we see that with the Goodnet topology, the sudden increase in QoS dropped flows occurs earlier at 1500 flows. This is because Goodnet is sparser than AttMpls (Refer to Figure 5), which leads to faster exhaustion of rule-capacity. However, the proposed scheme takes into account the rule-capacity of the switches and thus achieves significant improvement in terms of QoS violated flows when the number of flows is large. Thus, the proposed scheme is scalable to a large number of flows and is suitable for an IoT environment consisting of a huge number of devices.

Figure 10 shows the percentage of QoS violated flows with varying ratio of *ds* and *ls* flows. We observe that although the performance of the proposed scheme varies with different ratio of *ds* and *ls* flows, the proposed scheme outperforms the benchmark schemes in all the cases. Hence, the proposed scheme is capable of maintaining good performance with varying ratios of delay-sensitive and loss-sensitive flows. Therefore, it is suitable for an IoT environment.

5.4.4 Activated Links

We evaluate the percentage of links activated in the network. Figure 11 shows the percentage of links activated using the proposed scheme, Sway, compared to the existing benchmark schemes with different number of flows. We observe that Sway has a higher percentage of link activation

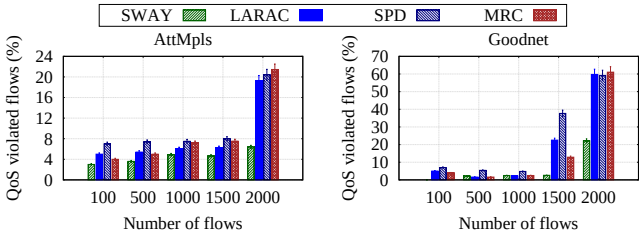


Figure 9: Percentage of QoS violated flows in the network with varying number of flows

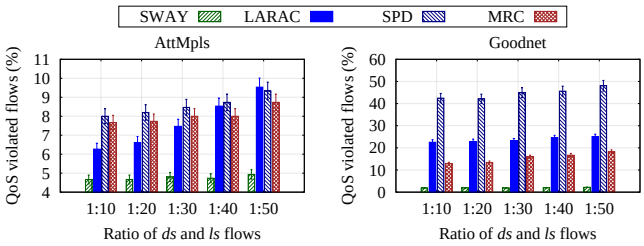


Figure 10: Percentage of QoS violated flows in the network with varying ratio of *ds* and *ls* flows (1500 flows)

compared to the benchmark schemes, since it chooses optimal forwarding paths for both the *delay-sensitive* and *loss-sensitive* flows, by constructing different minimum spanning trees, dependent on the *type* of each flow. Further, the rule-capacity constraints of SDN switches also contribute to increased link activation, as discussed in Section 3.4. In particular, Sway incurs 13%, 15%, and 13% (with AttMpls topology) and 10%, 13%, and 10% (with Goodnet topology) more link activation (on average) as compared to LARAC, SPD, and MRC schemes.

From the above analysis, we see that the proposed scheme is capable of achieving optimal routing in terms of end-to-end delay and the number of QoS violated flows in the network, at the cost of increased link activation. The proposed scheme achieves optimal routing by making the best use of the link characteristics, and the type and behavior of the flows along with their associated requirements.

6 Conclusion

In this paper, we proposed a traffic-aware QoS routing scheme in SDIoT, which takes into account the differing QoS requirements of heterogeneous flows. We considered the heterogeneous flows as either delay- or loss-sensitive. We proposed a greedy approach based on Yen's K-shortest paths algorithm to calculate feasible routing path. Simulation results showed that compared to the existing single metric routing schemes, the proposed scheme can provide QoS to both delay- and loss-sensitive flows. Results also indicated that the proposed scheme significantly reduces the end-to-end delay and number of QoS violated flows in the network, while maintaining the QoS requirements.

In this work, we assumed that flows are present in the network are either — delay- or loss-sensitive or both. However, in a large-scale network, it is expected that multiple heterogeneous devices will participate. Therefore, we plan

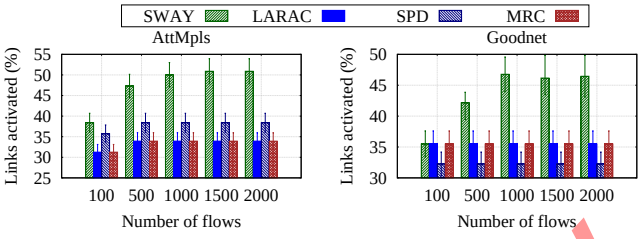


Figure 11: Percentage of links activated in the network

to consider other type of flows (such as jitter-sensitive), in the network, as the future extension of this work. Additionally, we also plan to propose a low-level packet classification scheme for fine-grained QoS forwarding.

References

[1] C. Perera, C. H. Liu, and S. Jayawardena, "The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 4, pp. 585–598, 2015.

[2] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1457–1477, 2017.

[3] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, "Network Architecture and QoS issues in the Internet of Things for a Smart City," in *Proc. Int. Symp. Communications and Information Technologies*, 2012, pp. 956–961.

[4] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A Software Defined Networking architecture for the Internet-of-Things," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[5] L. Li, S. Li, and S. Zhao, "QoS-Aware Scheduling of Services-Oriented Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1497–1505, 2014.

[6] Cisco Systems Inc., "The Zettabyte Era: Trends and Analysis," *White Paper, Cisco Visual Networking*, 2014.

[7] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN)," Protocol Specification Version 1.2, 2013.

[8] Z. Shelby, K. Hartke, and C. Bormann, " The Constrained Application Protocol (CoAP)," Internet Requests for Comments, RFC Editor, RFC 7252, 2014.

[9] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, 2017.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[11] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 710–715, 2013.

[12] 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.

[13] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the Effect of Forwarding Table Size on SDN Network Utilization," in *Proc. IEEE INFOCOM*, 2014, pp. 1734–1742.

[14] H. Huang, S. Guo, P. Li, B. Ye, and I. Stojmenovic, "Joint Optimization of Rule Placement and Traffic Engineering for QoS Provisioning in Software Defined Network," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3488–3499, 2015.

[15] Y. Liu, D. Niu, and B. Li, "Delay-Optimized Video Traffic Routing in Software-Defined Interdatacenter Networks," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 865–878, 2016.

[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] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh. (2016) SDFog: A Software Defined Computing Architecture for QoS Aware Service Orchestration over Edge Devices. [Online]. Available: arXiv:1609.01190

[19] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for iot," *Springer Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.

[20] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martnez, T. Tsuritani, and I. Morita, "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," *IEEE/OSA Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1420–1428, 2018.

[21] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing Wireless Fog Networks using Software-Defined Networking," in *Proc. IEEE/ACS Int. Conf. Computer Systems and Applications*, 2017, pp. 1149–1156.

[22] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An Intelligent Traffic Load Prediction Based Adaptive Channel Assignment Algorithm in SDN-IoT: A Deep Learning Approach," *IEEE Internet of Things Journal*, 2018, doi: 10.1109/JIOT.2018.2838574.

[23] P. Bellavista, C. Giannelli, T. Lagkas, and P. Sarigiannidis, "Quality Management of Surveillance Multimedia Streams Via Federated SDN Controllers in Fiwi-Iot Integrated Deployment Environments," *IEEE Access*, vol. 6, pp. 21 324–21 341, 2018.

[24] H. Li, M. Dong, and K. Ota, "Control Plane Optimization in Software-Defined Vehicular Ad Hoc Networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7895–7904, 2016.

[25] K. Sood, S. Yu, and Y. Xiang, "Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review," *IEEE Internet of Things J.*, vol. 3, no. 4, pp. 453–463, 2016.

[26] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[27] "A Novel Non-supervised Deep Learning Based Network Traffic Control Method for Software Defined Wireless Networks," *IEEE Wireless Communications*, 2018, doi: 10.1109/MWC.2018.1700417.

[28] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IOT," in *Proc. IEEE Int. Conf. Cyber, Physical and Social Computing*, 2011, pp. 717–720.

[29] I. Awan, M. Younas, and W. Naveed, "Modelling QoS in IoT Applications," in *Proc. of the IEEE Int. Conf. Network-Based Information Systems*, 2014, pp. 99–105.

[30] 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 Workshops*, 2017, pp. 559–564.

[31] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," Internet Requests for Comments, RFC Editor, RFC 7228, 2014.

[32] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled Software Sefined Networking for Efficient and Scalable IoT Communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, 2015.

[33] H. Guo, J. Liu, Z. Fadlullah, and N. Kato, "On minimizing energy consumption in fiwi enhanced lte-a hetnets," *IEEE Trans. Emerg. Topics Comput.*, no. 99, 2016, doi: 10.1109/TETC.2016.2598478.

[34] H. Guo, J. Liu, and L. Zhao, "Big data acquisition under failures in fiwi enhanced smart grid," *IEEE Trans. Emerg. Topics Comput.*, 2017, doi: 10.1109/TETC.2017.2675911.

[35] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things J.*, vol. 1, no. 2, pp. 112–121, 2014.

[36] H. Sawashima, Y. Hori, H. Sunahara, and Y. Oie, "Characteristics of UDP packet loss: Effect of tcp traffic," in *Proc. Annu. Conf. of the Internet Society*, 1997.

[37] J. F. Huang, G. Y. Chang, C. F. Wang, and C. H. Lin, "Heterogeneous flow table distribution in software-defined networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 252–261, 2016.

[38] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, "FlowQoS: QoS for the Rest of Us," in *Proc. ACM Workshop Hot Topics in Software Defined Networking*, 2014, pp. 207–208.

[39] B. Ng, M. Hayes, and W. K. G. Seah, "Developing a traffic classification platform for enterprise networks with SDN: Experiences amp; lessons learned," in *IFIP Networking Conf.*, 2015, pp. 1–9.

[40] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *Proc. IEEE NOMS*, 2014, pp. 1–8.

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

[42] Nokia, "An internet of things blueprint for a smarter world," Nokia, Tech. Rep., 2014.

[43] R. Ratasuk, N. Mangalvedhe, A. Ghosh, and B. Vejlgaard, "Narrowband LTE-M System for M2M Communication," in *Proc. IEEE Vehicular Technology Conf.*, 2014, pp. 1–5.

[44] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *Proc. ACM SIGCOMM Workshop Hot Topics in Networks*, 2010, pp. 19:1–19:6.

[45] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[46] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com