2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) | 978-1-6654-4935-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/MASS52906.2021.00082

A QoS-aware Routing based on Bandwidth Management in Software-Defined IoT Network

Priyanka kamboj
Department of Computer Science and Engineering
Indian Institute of Technology Ropar
2018csz0003@iitrpr.ac.in

Sujata Pal
Department of Computer Science and Engineering
Indian Institute of Technology Ropar
sujata@iitrpr.ac.in

Ambika Mehra
Department of Computer Science and Engineering
Chitkara University, India
amehra.cse17@chitkarauniversity.edu.in

***Abstract*—The burgeoning demands of the Internet of Things (IoT) applications such as video/audio streaming in surveillance, disaster recovery, multimedia, and healthcare paves the need to provide better Quality of Service (QoS) delivery. The growth in the amount of data generated by multimedia applications increases congestion in the network. Software-Defined Networking (SDN) is an emerging approach that centrally controls the network and solves the congestion problem in the network. SDN has many advantages that help in network management, traffic shaping, and routing in the IoT network to manage the data generated from a diverse range of applications. In this paper, we propose a congestion technique using Hierarchical Token Bucket (HTB) to manage the bandwidth in the Software-Defined IoT (SDIoT) network. Further, we propose a routing scheme to compute optimal routing shortest paths using Dijkstra's algorithm by selecting the *min-cost* path based on the priorities of traffic flows. The results illustrate that the proposed approach achieves a reduction in end-to-end delay by 38%, 44% and higher average throughput by 29%, 43% in comparison with the benchmark schemes – SDN with HTB and the Delay Minimization method, respectively.**

***Keywords*-Congestion, Internet of Things, Quality of Service, Routing, Software Defined Networking**

I. Introduction

The evolution of networks leads to the discovery of the Internet of Things (IoT) which tends to interconnect people, devices, and different objects together [1]. The heterogeneous network interconnects the smart objects such as laptops, sensors, home appliances, cars, radio-frequency identification tags, smartphones, and access points [2]. The present Internet architecture due to its distributed nature contributes towards the interconnection of these devices/objects to provide access to a wide range of communication technologies [3]. The number of devices using the Internet is rising at a startling rate which creates a demand for resources such as bandwidth on the heterogeneous constrained backbone network [4].

A. Motivation

The different IoT applications in various sectors such as multimedia, entertainment, healthcare, agriculture, and autonomous vehicles demand better service delivery to guarantee Quality of Service (QoS) to end-users [1]. The current Internet architecture provides best-effort service delivery [4], [5]. Some of the multimedia-based applications are delay sensitive while others are packet loss sensitive [6]. Due to the heterogeneity in the IoT network, it fails to satisfy the QoS guarantee to the end-user applications. Software-Defined Networking (SDN) is an emerging technology that provides flexibility into the network due to its programmable nature [7]. Due to the abstraction in SDN, it allows the dynamic configuration of the application-level policies and addresses the limitations of current Internet architecture [8]. Therefore, SDN in IoT networks (SDIoT) provides an ability to adhere to problems of various high-level applications and imparts to enhance QoS to the end-users.

B. Contribution

In this paper, we propose an SDN-based approach that considers the QoS requirements of traffic flows in the IoT network. In particular, our proposed approach varies from the state-of-the-art in two ways. Firstly, we introduce a congestion technique using Hierarchical Token Bucket (HTB) traffic shaping algorithm in an SDIoT network to manage the bandwidth in the network. The proposed architecture utilizes the programming nature of SDN and satisfies the end-users QoS requirements by considering link congestion in the network. Secondly, we propose a routing scheme to compute *min-cost* routing paths for the traffic flows in the network.

The contribution of the work is stated as follows:

1) We propose an approach that uses the HTB queuing discipline to manage the bandwidth in the SDIoT network.
2) We propose a routing scheme to compute optimal routing paths considering congestion in the network while taking into account QoS requirements of TCP and UDP traffic flows.
3) The proposed routing scheme computes the shortest routing path using the Dijkstra algorithm and selects

978-1-6654-4935-9/21/$31.00 ©2021 IEEE
DOI 10.1109/MASS52906.2021.00082

the *min-cost* path based on the priority of traffic flows in the network.

4) The proposed architecture has been evaluated using the Mininet emulator and POX controller used in SDN. The proposed approach achieves a reduction in end-to-end delay by 38%, 44% and higher average throughput by 29%, 43% in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method, respectively.

C. Organization

We organize the paper according to different sections. Section II discusses the state-of-the-art. Section III describes the system architecture of the network. Section IV illustrates the proposed approach and routing algorithm used in the SDIoT network. Section V describes the performance metrics used in the experiment and discusses the obtained results. Finally, Section VI consists of the conclusion and future work.

II. Related Work

In this section, we address the state-of-the-art that has been considered by various researchers to improve the QoS in SDN-based IoT networks.

Sun *et al.* [9] studied the problem of intelligent routing of traffic flows generated by IoT applications. The different types of traffic flows in IoT applications have different QoS requirements. Thus, to guarantee QoS requirements of traffic flows, the authors introduced a data flow classification algorithm to classify the data flow and route them on suitable paths. Llopis *et al.* [10] introduced a delay minimization method for routing traffic flows in SDN. The authors have considered only metric delay but not considered bandwidth utilization while routing traffic flows in the network. Deng *et al.* [8] proposed an application-aware QoS routing algorithm to provide routing paths to traffic in an SDN-based IoT network. The authors failed to consider the priority-based transmission from end-user applications.

Ren *et al.* [11] utilize the concept of Hierarchical Token Bucket (HTB) to manage the bandwidth by using the borrowing technique. The best-effort flows are routed on the shortest path using the Dijkstra algorithm. The authors failed to evaluate the performance of their proposed approach on QoS metrics such as delay and throughput. Further, the authors in [12] proposed an HTB queuing discipline to guarantee QoS for DiffServ provisioning. The results had been validated to dynamically optimize the bandwidth sharing by redistributing the unused bandwidth to the backlogged flows. But they have not taken into account the priority-based multimedia transmission to enhance QoS.

The state-of-the-art reveals that a research lacuna exists to consider the traffic flows based on the priority of real-time applications. The existing literature work approaches considered constraints to manage the bandwidth in the network for routing of traffic flows. But the authors have not taken into account the use of HTB to manage bandwidth by considering the priority of real-time application traffic flows in SDIoT networks.

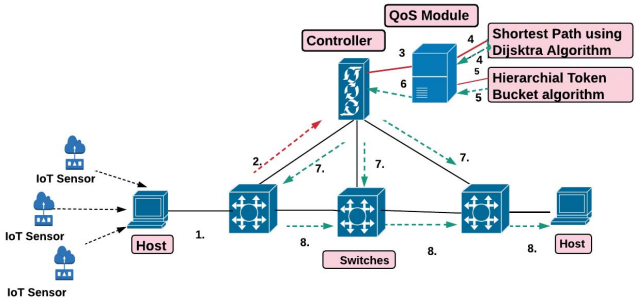


Figure 1: Proposed System Architecture using Hierarchical Token Bucket algorithm

III. System Model

This section presents the proposed system architecture of the SDIoT network as shown in Figure 1.

A. Architecture

In the proposed architecture, the host sends the traffic flows to the switch linked with the host in the network. The switches are connected to the SDN controller by the OpenFlow API as shown in Figure 1. The application layer consists of a QoS module deployed on the top of the controller. The QoS module comprises two main sub-modules – the first is the computation of the shortest path using the Dijkstra algorithm and the second is queuing discipline in which HTB is used to regulate the bandwidth in the SDIoT network. On the arrival of a new packet from a host, the switch checks whether its flow entry exits in the flow table. If no such entries are there in the flow table, it then sends the Packet-in to the controller. The controller takes necessary actions such as add, drop, or delete the flow entries in the flow table for the incoming Packet-in message. The controller decides with the help of a QoS module that is deployed in the form of an application on top of the POX controller.

Let us consider the SDN network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$. Here, $\mathcal{V}$ represents the set of OpenFlow switches and

$$\mathcal{L} = \{(s,t)|s,t \subset \mathcal{V}, s \neq t\}, \quad (1)$$

denotes the set of links between the switches that is the link from switch i to switch j as shown in Equation 1. Here, s and t represent the source and destination switch.

B. Impact of QoS on IoT applications

The demand for QoS in the IoT applications is present at all the layers – the physical layer, link layer, network layer, and transport layer. The IoT applications can use

either User Datagram Protocol (UDP) or Transport Control Protocol (TCP) at the transport layer. TCP is a connection-oriented protocol due to which it extrapolates the latency as each session starts with the connection setup phase. Broadly, UDP is most applicable for the applications that are delay-sensitive because it is connection-less oriented. Presently, the Internet traffic flows drifts by TCP communication protocols for applications, such as video streaming which severely affect the QoS. Therefore, it becomes necessary to build a competent end-to-end path for the efficient routing of the distinct traffic flows in the network.

The different types of flows in the network can be represented mathematically as:

$$\mathcal{F} = \{f_z \mid z \, \epsilon \, N\}, f_z := (s_z, t_z, \rho_z, \beta_z), \tag{2}$$

where, s_z, t_z, ρ_z , β_z represents the source, destination, type of the flow (UDP or TCP), bandwidth demand of traffic flow.

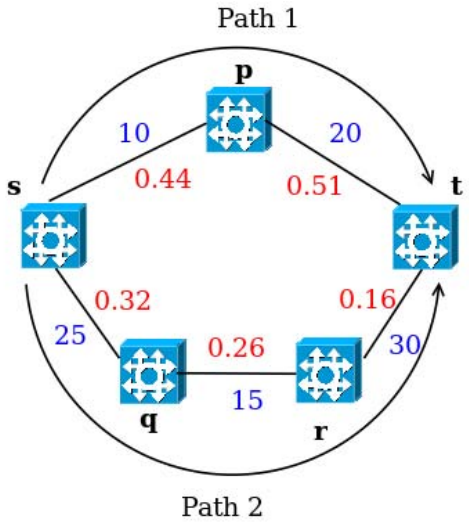


Figure 2: Illustration of Path 1 and Path 2 for routing the traffic flows from source to destination

Each link is associated with a bandwidth capacity $c(i, j)$ and delay $d(i, j)$ between nodes i and j. In Figure 2, s is the source switch, t is the destination switch and p, q, and r are the intermediate nodes/switches. Let $P_{(s,\,t)}$ represent the set of paths in the network topology. The set of the paths is defined as:

$$P_{(s,\,t)} = \{p_i \mid p_i \; is \; a \; path \; from \; (s,t)\}\,, \; i = \{1, 2, ...n\}\,. \tag{3}$$

In Figure 2, there are two paths represented as Path 1 and Path 2. The shortest path is calculated according to its bandwidth requirements for a UDP or TCP traffic flow in an IoT application.

IV. Proposed Routing Approach

In this section, we compute the routing paths while considering the QoS requirements of traffic flows. The objective is to find the *min-cost* routing path that fulfills the essential bandwidth demand of the flows while considering the priority of traffic flows.

A. QoS-aware routing in Software-defined IoT network

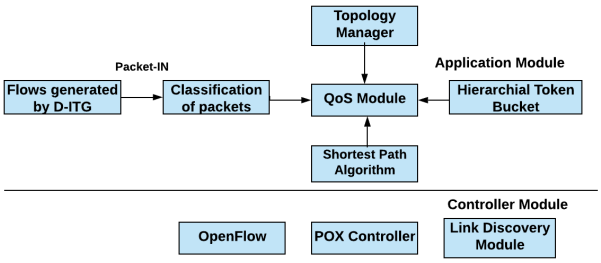


Figure 3: Proposed architecture represents the different modules in the controller layer

Figure 3 represents the proposed controller architecture. The D-ITG traffic generator [13] generates the UDP/TCP traffic flows. The Packet-in module of the controller handles the Packet-in message generated by the switches. However, the **classification module** classifies the packet based on the type of packet generated as Address Resolution Protocol (ARP), Link Layer Discovery Protocol (LLDP), Internet Protocol (IP). The classification module also assigns the priority level to the hosts. The switches forward the Packet-in messages that arrive at them to the controller. The proposed architecture uses the POX controller, and the OpenFlow 1.0 protocol provides communication between the switches and the controller. As the mesh topology starts running, the **link discovery module** discovers the links that generate between the hosts and the switches. The **topology manager module** represents an overall scenario of the network to the QoS module. The **QoS module** calculates the shortest path using the Dijkstra algorithm and uses the HTB queue discipline algorithm. Keeping this concept in mind, Algorithm 1 revolves around the idea of communication between the host and destination switch and finds the shortest path between the source and the destination node using the cost function stated in Equation 4. It also uses HTB to assign queue ID (as qid) to the links between host and switch for managing the link bandwidth in Algorithm 1.

Further, the classification module checks the bandwidth demand of the incoming flows and classifies the packets based on their priority. The QoS module computes the cost of all the paths using Equation 4 where, $c(i, j)$ and $d(i, j)$ represents the bandwidth and delay of a link (i, j) respectively, whereas, $c_{i,j}^{util}$ denotes the link capacity utilization and d_f^{max} is the maximum delay a flow can tolerate. The parameter α is user-defined to measure the impact of these factors on the cost function. Further, the module sorts the paths based on the ascending order of the cost. The

Algorithm 1 Routing Algorithm

Input : Graph $\mathcal{G}$, source and destination switch represented as src_switch, dst_switch, Set of flows $\mathcal{F}$ ▷ User defined

Output : Path on which flow is routed

```
function HANDLE_PACKET_IN(src_switch)
    if pkt.type == ARP then ▷ new Packets are arrived at controller
        for all Ip's in [switch_list] do
            if destIp ← [switch_list] then
                src_switch → dest.MAC
                src_switch ← dest.IP
            else
                flooding_on_all_switches (pkt)
            end if
        end for
    end if
    if pkt.type == IP then
        a = IPv4.payload
        if src_switch is TRUE then
            src_switch → qid(dest_host)
            if src_switch.dpid == dest.dpid then
                install_path(dest_dpid, src_switch)
            else
                Dijkstra_algo(src_switch, dest_switch)
                install_path(dest_dpid, src_switch) ▷ install_path function installs the routing paths in switches
            end if
        end if
    end if
end function
```

controller checks the priority of the flows and assigns the highest priority flow to the *min-cost* path. Thus, the different traffic flows can be forwarded through paths to minimize the congestion in the network. Finally, the cost of the paths gets updated as the residual bandwidth of the links is updated.

$$\Phi(i,j) = \alpha \frac{c_{i,j}^{util}}{c_{i,j}} + (1-\alpha)\frac{d(i,j)}{d_{i,j}^{max}}. \quad (4)$$

We analyze the running time complexity of our proposed Algorithm 1. The Dijkstra algorithm computes the shortest paths between the source and the destination node. The running time complexity of Dijkstra function in case of adjacency list representation in Graph $\mathcal{G}$ is $O(|\mathcal{V}| + |\mathcal{L}|\, log\, |\mathcal{V}|)$. The Dijkstra function is run for $|\mathcal{F}|$ times. Thus, the execution time for our proposed algorithm is $O\,|\mathcal{F}| * (|\mathcal{V}| + |\mathcal{L}|\, log\, |\mathcal{V}|)$.

B. Network Topology

We consider a mesh topology that consists of switches represented as s1, s2, s3, and s4, and hosts as H1, H2, H3, H4, and H5 for the experiment as shown in Figure 4. The host H1 acts as the server. The Distributed Internet Traffic generator (D-ITG) [13] generates the IoT traffic flows in the network topology. Table I illustrates the flow specification of the different traffic flows generated from hosts H3, H4, and H5. Each link in the network operates at 1 Gbps. The network topology uses the approach of queue discipline that is HTB algorithm. The HTB performs traffic shaping using the token bucket filter algorithm [12]. In the topology, server H1 has two queues represented as queue 1 and queue 2. Further, the bandwidth has been set at 6 Mbps and 4 Mbps between the server (H1) and switch s1. The identities to queues are assigned to differentiate between them. When the host H3 sends the UDP or TCP traffic flows to server H1, the traffic flows will enter into queue 1. On the other hand, the incoming traffic flows from host H4 and H5 will arrive in queue 2. Thus, it helps to manage the bandwidth between the different hosts and the switches to reduce the congestion in the network.

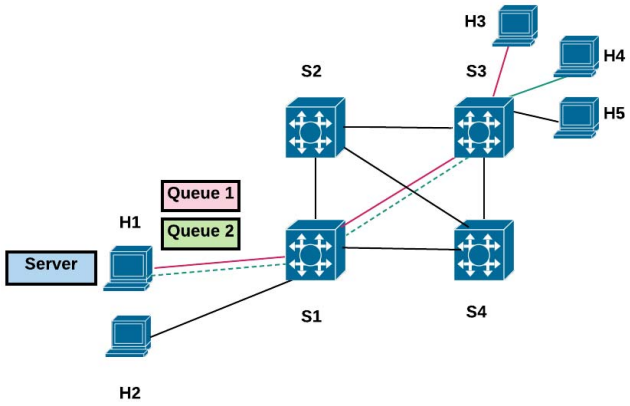


Figure 4: Network topology used in the work

Table I: Flow specification of distinct services

Flows	Hosts	Protocol	Priority	Bandwidth
F1	H3	TCP	1	20 Mbps
F2	H4	TCP	2	30 Mbps
F3	H5	UDP	3	40 Mbps

V. Performance Evaluation

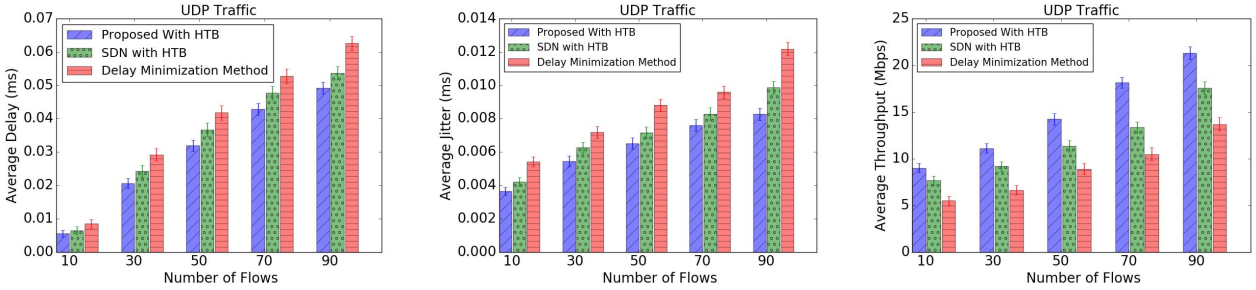
A. Simulation Settings

The proposed topology has been generated using the Mininet emulator [13] and the SDN POX[1] controller. We experimented using the HP-ProDesk using Intel Core i7 CPU, 3.60 GHz processors, and 8 GB RAM.

B. Results and Discussion

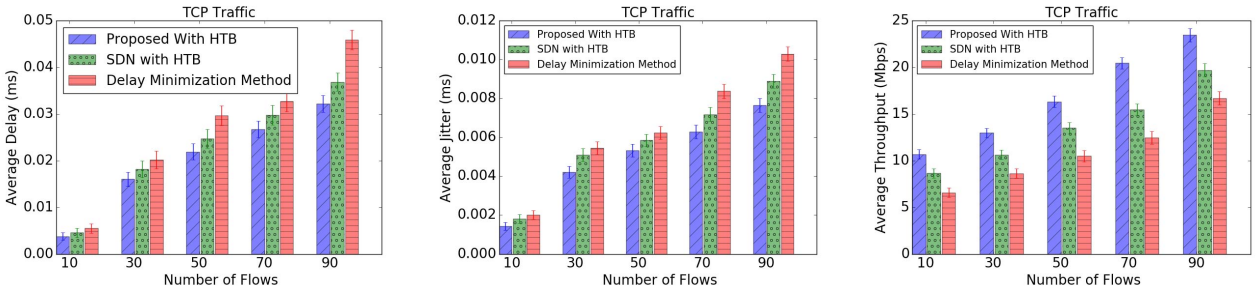
The experiment is performed by generating the traffic flows such as UDP and TCP flows using the D-ITG traffic

[1]https://github.com/noxrepo/



(a) End-to-end delay (b) Average jitter (c) Average throughput

Figure 5: End-to-End delay, average jitter and average throughput with varying number of flows using UDP traffic



(a) End-to-end delay (b) Average jitter (c) Average throughput

Figure 6: End-to-End delay, average jitter and average throughput with varying number of flows using TCP traffic

generator. In our work, we consider different metrics such as end-to-end delay, average jitter, and average throughput to ensure better QoS delivery to the applications. We have performed a comparison of the proposed approach with benchmark schemes – SDN with HTB [11] and the Delay Minimization method [10]. The experiment is carried out 30 times to calculate the average value for each parameter. We have computed the results at 90% confidence intervals to compute the average for every metric. The number of packets propagated in the case of 10, 30, 50, 70, 90 flows are 10,000, 30,000, 50,000, 70,000, 90,000 respectively.

1) End-to-End Delay: We computed the average delay for each path with the increasing number of traffic flows in the network. Figure 5a and Figure 6a illustrates the end-to-end delay using UDP and TCP traffic flows using mesh topology respectively. We analyzed that the proposed approach achieves a reduction in end-to-end delay by 26%, 37% (with TCP flows), and 38%, 44% (with UDP flows) in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method. We noticed that the proposed approach results in lesser delay in UDP traffic and higher in TCP traffic with the increase in the number of flows. The reason is that TCP is connection-oriented and traffic flows take more time in establishing the connection setup phase and results in more end-to-end delay. However, the proposed approach achieves lesser end-to-end delay as it assigns the higher priority traffic flows to the least cost path and thus, reduces congestion in the network. On the other hand, SDN with HTB scheme results in lesser delay as compared to the Delay Minimization method. The approach uses the HTB queue discipline and computes the shortest path for all traffic flows without considering delay and bandwidth requirements of flows, thus results in congestion and also increases the delay in the network. However, in the Delay Minimization method, the algorithm searches for a path with minimum delay using the brute-force approach. This method does not consider link capacity utilization to route the traffic flows and results in higher delay.

2) Average Jitter: Figure 5b and Figure 6b represents average jitter using UDP and TCP traffic flows in network topology respectively. We analyzed that the proposed approach achieves a reduction in average jitter by 13%, 19% (with TCP flows), and 15%, 24% (with UDP flows) in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method. The proposed scheme considers link congestion in the network and redirects the traffic flows with the highest priority to the least cost path. We noticed that the proposed approach results in lesser average jitter in TCP and UDP traffic with the increasing number of flows in the network. On the other hand, SDN

with HTB scheme has not considered jitter while routing the traffic flows in the network topology. It results in less jitter as compared to the Delay Minimization method. On the contrary, the Delay Minimization method considered only a single metric delay and results in higher jitter. However, our proposed approach performs better than the benchmark schemes as it considered the delay variation while performing traffic forwarding in the network.

3) Average Throughput: Figure 5c and Figure 6c represents the throughput achieved using both TCP and UDP traffic flows respectively. We analyzed that the proposed approach achieves higher throughput by 31%, 51% (with TCP flows), and 29%, 43% (with UDP flows) in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method. We observed that with the increase in the number of flows, throughput while using the proposed approach is higher than the benchmark schemes. The reason is that the proposed approach sends the traffic flows on the least-cost path based on the priority of traffic flows and reduces congestion in the network. On the other hand, SDN with HTB scheme does not consider the bandwidth requirements of the traffic flows and results in lesser network throughput. However, the Delay Minimization method does not find the optimal routing path while satisfying the QoS requirements of the traffic flows. Thus, both the traffic flows achieve lesser throughput in the network in comparison to the proposed scheme.

VI. Conclusions

In this paper, we proposed a congestion technique to manage the bandwidth in a Software-Defined IoT network. We formulate a routing problem to find optimal routing paths for different traffic flows (TCP or UDP) generated using a D-ITG generator. The routing scheme computes the shortest path using Dijkstra's algorithm by selecting the *min-cost* path based on the priorities of traffic flows. We experimented using different QoS parameters such as end-to-end delay, jitter, and throughput. The results demonstrate the effectiveness of the proposed approach. The results signify that the proposed approach achieves a reduction in end-to-end delay by 26%, 37% (with TCP flows), and 38%, 44% (with UDP flows) in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method, respectively. Further, the proposed approach achieves higher average throughput by 31%, 51% (with TCP flows), and 29%, 43% (with UDP flows) in comparison to the benchmark schemes – SDN with HTB and the Delay Minimization method, respectively. The single-path routing leads to congestion in the network. Therefore, as a future expansion of this work, we will use multiple path routing to deal with congestion in the network.

References

[1] P. Kamboj and S. Pal, "QoS in Software Defined IoT Network using Blockchain Based Smart Contract," in *Proc. of the* 17^{th} *Conf. on Embedded Networked Sensor Systems*, (New York, USA), pp. 430–431, Nov. 2019.

[2] F. Naeem, M. Tariq, and H. V. Poor, "SDN-Enabled Energy-Efficient Routing Optimization Framework for Industrial Internet of Things," *IEEE Trans. on Ind. Informat.*, vol. 17, no. 8, pp. 5660–5667, 2021.

[3] A. N. Abbou, T. Taleb, and J. Song, "A Software-Defined Queuing Framework for QoS Provisioning in 5G and Beyond Mobile Systems," *IEEE Network*, vol. 35, no. 2, pp. 168–173, 2021.

[4] J. L. Herrera, J. Galán-Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing the Response Time in SDN-Fog Environments for Time-Strict IoT Applications," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[5] P. Kamboj and G. Raj, "Analysis of Role-Based Access Control in Software-Defined Networking," in *Proc. of* 5^{th} *Int. Conf. on Soft Comput. for Problem Solving*, (India), pp. 687–697, Springer, March 2016.

[6] 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 *Conf. on Computer Commun. Workshops (INFOCOM WKSHPS)*, (Atlanta, GA, USA), pp. 559–564, IEEE, May 2017.

[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-defined Networking: Past, Present, and Future of programmable networks," *IEEE Commun. Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[8] G.-C. Deng and K. Wang, "An Application-aware QoS Routing Algorithm for SDN-based IoT Networking," in *Symposium on Computers and Commun.*, (Natal, Brazil), pp. 186–191, IEEE, June 2018.

[9] W. Sun, Z. Wang, and G. Zhang, "A QoS-guaranteed intelligent routing mechanism in software-defined networks," *Computer Netw.*, vol. 185, p. 107709, 2021.

[10] J. M. Llopis, J. Pieczerak, and T. Janaszka, "Minimizing Latency of Critical Traffic through SDN," in *Int. Conf. on Networking, Architecture and Storage*, (Long Beach, CA, USA), pp. 1–6, IEEE, August 2016.

[11] S. Ren, Q. Feng, and W. Dou, "An End-to-End QoS Routing on Software Defined Network Based on Hierarchical Token Bucket Queuing Discipline," in *Proc. of the Int. Conf. on Data Mining, Commun. and Inform. Technology*, (New York, USA), pp. 1–5, May 2017.

[12] C.-H. Lee and Y.-T. Kim, "QoS-aware Hierarchical Token Bucket (QHTB) Queuing Disciplines for QoS-guaranteed Diffserv provisioning with optimized bandwidth utilization and priority-based preemption ," in *Proc. of the Int. Conf. on Inform. Networking*, (Bangkok, Thailand), pp. 351–358, January 2013.

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