ReSQ: Reinforcement Learning-Based Queue Allocation in Software-Defined Queuing Framework

Ilora Maity[1], and Tarik Taleb[2]
[1]Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg, (email: ilora.maity@uni.lu)
[2]Information Technology and Electrical Engineering, University of Oulu, Oulu, Finland and Department of Computer and Information Security, Sejong University, Seoul, South Korea (email: tarik.taleb@oulu.fi)

With the evolution of 5G networks, the demand for Ultra-Reliable Low Latency Communications (URLLC) services is increasing. Software-Defined Networking (SDN) offers flexible network management to prioritize URLLC services coexisting with best-effort traffic. Utilizing the network programmability feature of SDN, Software-Defined Queueing (SDQ) framework selects the optimal output port queue on forwarding devices and routing path for incoming traffic flows to provide deterministic Quality of Service (QoS) support required for URLLC traffic. However, in the existing SDQ framework, the selections of optimal queue and path are done manually by observing the traffic type of each incoming flow, the available bandwidth of each potential routing path, and the status of output port queues of each forwarding device on each potential routing path. The static allocations of path and queue for each flow are inefficient to provide a deterministic QoS guarantee for a high volume of incoming traffic which is typical in 5G networks. The limited buffer availability on the forwarding devices is another constraint regarding optimal queue allocation that ensures an end-to-end (E2E) delay guarantee. To address these challenges, in this paper, we extend the SDQ framework by automating queue management with a reinforcement learning (RL)-based approach. The proposed queue management approach considers diverse QoS demands as well as a limited buffer on the forwarding devices and performs prioritized queue allocation. Our approach also includes a hash-based flow grouping to handle a high volume of traffic having diverse latency demands and a path selection mechanism based on available bandwidth and hop count. The simulation result shows that the proposed scheme ReSQ reduces the QoS violation ratio by 10.45% as compared to the baseline scheme that selects queues randomly.

Index Terms—**SDN, Reinforcement Learning, QoS, Queue Allocation, Deterministic QoS, URLLC**

I. Introduction

Ultra-Reliable Low Latency Communication (URLLC) constitutes a significant portion of 5G communication services. URLLC services such as critical healthcare applications, emergency services for vehicular networks, and use cases for the Industrial Internet of Things (IIoT) have stringent latency requirements beyond the capability of traditional networks [20]. Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet) are two emerging paradigms that provide support for URLLC [1]. TSN is a set of standards for Layer 2 that provide a deterministic Quality of Service (QoS) guarantee [2]. DetNet extends the policy of TSN for Layer 3 and selects routing paths for ensuring deterministic QoS support. Software-Defined Networking (SDN) offers programmable networking to realize the queuing algorithms with configurable parameters defined in TSN standards and deterministic QoS support [3] [4] [5]. Software-Defined Queueing (SDQ) [6] expands the idea of deterministic QoS provisioning by providing optimal routing paths and selecting specific priority queues on SDN switches. Additionally, SDQ performs real-time queue management by adding, deleting, or updating existing queues. However, the selection of routing path and queue is fixed in SDQ which is not suitable for networks where the traffic is heterogeneous in terms of requirements and volume.

Corresponding author: Ilora Maity (email: ilora.maity@uni.lu).

Deterministic QoS support for URLLC traffic depends on both the selected switches to route the traffic and the right queues the traffic should traverse at each switch in the routing path. Therefore, route and queue management at the granular level is necessary. Several existing works attempt to achieve this granularity in route and queue selection utilizing the network programmability feature of SDN. Tomovic *et al.* [12] proposed an SDN-based QoS provisioning framework that maintains a dedicated queue for each priority flow. Yan *et al.* [11] proposed an SDN-based QoS provisioning approach that assures bandwidth for each traffic type by analyzing multiple available routing paths. Dutra *et al.* [5] utilized the QoS support in Openflow to provide an E2E QoS guarantee to each flow. However, the existing approaches [12] [11] [5] do not consider the requirement of creating new queues when the existing queues are incapable of accommodating an incoming flow due to insufficient capacity or requirement mismatch. SDQ [6] addresses this issue by facilitating autonomous queue creation/deletion based on the dynamics of network traffic which is heterogeneous in 5G networks. However, the queue and route allocation are fixed in the SDQ framework and the existing allocation cannot be changed with varying network traffic which is typical in the next generation networks. Additionally, the SDQ solution does not consider the limited buffer size in SDN switches [21]. Considering these limitations, we argue that the existing SDQ framework can be enhanced with learning-assisted queue management.

In this work, we propose a reinforcement learning(RL)-based queue allocation (ReSQ) approach extending the SDQ framework. The proposed scheme ReSQ models queue setting in each SDN switch considering fixed buffer size. To handle high traffic load, ReSQ performs path and queue allocation for flow groups with similar demands. Additionally, in ReSQ, existing path and queue allocations are modified dynamically when the QoS demands are not fulfilled. ReSQ has four main modules — packet grouping, path selection, queue allocation, and verification. The packet grouping module clusters similar packets in multiple priority groups using a hash function. The path selection module selects the routing path for each priority group based on the available bandwidth and path length. The queue allocation module executes an RL model [7] to dynamically allocate queues to incoming traffic based on respective priorities. Finally, the verification module identifies the violating traffic for updating the corresponding queue allocation. Additionally, this module updates the queue setting. The primary contributions of our work are as follows:

- We formulate a hash function to group packets of similar flows in a single priority group. This reduces the computation complexity as path selection and queue allocation are performed at a time for all flows in a priority group.
- We design a path selection algorithm that assigns a communication path to the incoming flows in each priority group based on hop count and available bandwidth.
- We design an RL model to dynamically allocate queues to each priority group. This model considers the QoS demands of the traffic and reduces QoS violations and the number of changes in the existing queue allocation.
- Finally, we perform identification of violating flows and queue setting update as per requirement.

Simulation results depict that the proposed scheme reduces both the end-to-end (E2E) latency and the QoS violation ratio as compared to the random queue assignment.

The remainder of this paper is organized as follows. Section II discusses the SDQ framework and related works. Section III describes the system model. In Section IV, we discuss the proposed scheme. Section V depicts the simulation results. Finally, Section VI concludes the proposed work and discusses future research directions.

II. Related Works

In this section, we discuss RL-based traffic engineering, SDN-based solutions for providing QoS guarantees, and the SDQ framework which we extend in this work.

A. RL-Based Traffic Engineering

Several existing works propose RL-based traffic engineering in SDN. Zhang *et al.* [8] use RL to identify critical flows in SDN. Subsequently, the critical flows are rerouted to achieve better link utilization. Guo *et al.* [9] proposed a deep RL-based method to dynamically optimize the routing policy of traffic flows in Software-Defined Internet of Things networks based on historical data. Casas-Velasco *et al.* [10] presented an RL-based scheme that takes routing decisions based on link-state information in SDN.

B. SDN-Based Solutions for Providing QoS Guarantees

Various recent works investigate routing and resource management with softwarized network. Yan *et al.* [11] proposed an SDN-based QoS-aware scheme, named HiQoS, which explores multiple available routes and applies traffic-specific queuing algorithms to provide bandwidth guarantee. However, the QoS setting in HiQoS is fixed as compared to SDQ where queues are added or deleted based on the requirement. Tomovic *et al.* [12] proposed a framework for QoS provisioning in SDN. The proposed scheme dynamically routes flows based on priority and resource utilization. However, this approach uses fixed queues for each switch in the communication path of a priority flow. Guck *et al.* [4] presented two network models for deterministic QoS. In the first model, named the multi-hop model, each queue is associated with a rate and buffer budget. The second model, named threshold-based model considers a fixed maximum delay for each queue. However, our work considers dynamic queue allocation and queue setting update for QoS provisioning to heterogeneous traffic. Dutra *et al.* [5] introduced a QoS-aware resource management scheme for SDN-enabled networks. The proposed scheme aims to provide guaranteed E2E latency based on OpenFlow queue support. However, this work focuses on the reduction of resource over-utilization by minimizing the number of switches in the communication path. Goto *et al.* [24] proposed a queueing model for OpenFlow switches with individual queues for different classes of packets. Based on this model, the authors analyzed the average packet transfer delay and packet loss probabilities for different traffic classes. However, this model considers a single switch and a single controller only. Li *et al.* [23] modeled the delay of SDN-based avionics network considering 802.1Qbv switches and three traffic classes – best-effort, stream reservation, and control-data traffic. The authors proposed a queueing model at each switch where each traffic class is assigned to an individual first in first order (FIFO) queue. However, the objective of this work is to optimize bandwidth allocation. Rahouti *et al.* [22] proposed a priority-based queueing framework for SDN data and control planes to achieve QoS provisioning for heterogeneous traffic. In this approach, the data plane queue sizes are altered dynamically based on traffic demand, and the packet drop probability increases when the average queue size is beyond a threshold. This work considers finite priority levels and does not handle violating packets/flows.

C. SDQ Framework

Our work extends the SDQ framework proposed by Abbou *et al.* [6] for providing deterministic QoS support in SDN different from traditional queueing techniques. This framework is installed on top of the control plane. It has two basic components, namely traffic engineering and queue management. The traffic engineering module selects communication

paths for incoming packets based on available bandwidth. On the other hand, the queue management module allocates the packets to specific queues in the output ports. This module also performs the addition, deletion, modification, and scheduling of queues. SDQ is a promising framework that provides QoS guarantees that can benefit URLLC traffic which may co-exist with best-effort traffic. SDQ provides service-specific queueing and is appropriate for several use cases including the Industrial Internet of Things and Extended Reality. However, the performance of SDQ is limited in terms of high traffic load, re-allocation of routes and queues for under-performing flows, and heterogeneous traffic demands.

ReSQ enhances the SDQ framework by enabling dynamic queue allocation and queue setting updates for traffic flows having heterogeneous latency demands considering buffer limitation on SDN switches. Also, in this work, we group flows based on respective demands to handle the high traffic volume. Additionally, our work identifies the flows that violate the corresponding QoS demands and modifies queue assignments for these flows for better outcomes.

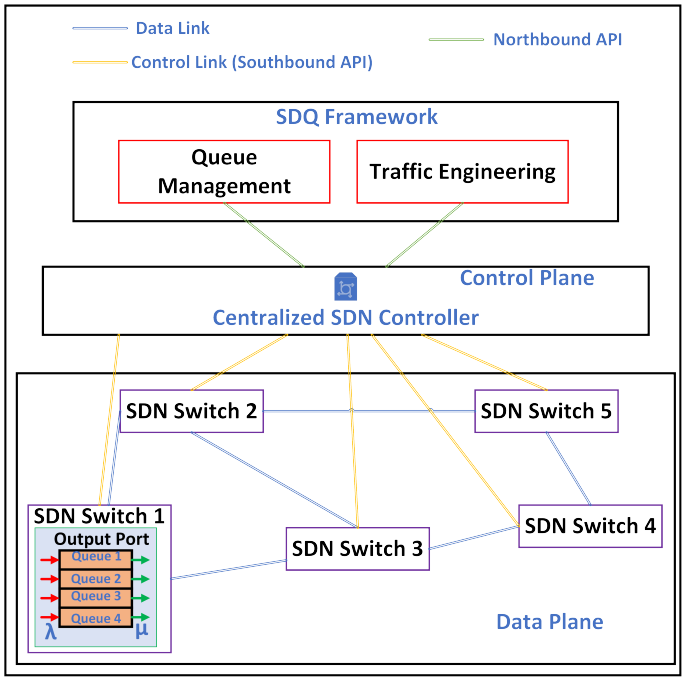


Fig. 1: Network topology with SDQ framework

III. System Model and Problem Formulation

As shown in Fig. 1, the network topology consists of a set of SDN switches denoted by N. We assume that the allocation of flows to queues is performed periodically. In each time slot, new flows and selected existing flows are assigned to selected queues. The allocation of a queue to a flow signifies that all packets belonging to the flow are assigned to the queue. Table I shows the symbols used in this paper.

TABLE I: Table of Symbols

Symbol	***Definition***
N	Set of switches
m_i	Number of queues associated with the output port of $n_i \in N$
q_j^i	The queue at j^{th} priority level at switch n_i
$\lambda_{i,j}(t)$	Packet arrival rate at q_j^i at time slot t
$\mu_{i,j}(t)$	Packet service rate at q_j^i at time slot t
$\rho_{i,j}(t)$	Traffic intensity at q_j^i at time slot t
K	Buffer size of a queue
$D_{i,j}(t)$	Average queueing delay of a packet assigned to q_j^i at time slot t
$F(t)$	Set of traffic flows at time slot t
$P_x(t)$	Set of packets belonging to $f_x \in F(t)$ at time slot t
D_x^{max}	Maximum allowable delay of $f_x \in F(t)$
$path_x(t)$	Ordered set of switches in the path of $f_x \in F(t)$ at time slot t
src_x	Source node of $f_x \in F(t)$
$dest_x$	Destination node of $f_x \in F(t)$
δ_x^k	Propagation delay between the k^{th} and $k+1^{th}$ switches in the set $path_x(t)$
$L_x(t)$	E2E latency of $f_x \in F(t)$

A. Queuing Model

Let a switch $n_i \in N$ has m_i queues associated with its output port. Each queue is assigned a priority level and based on the priority level the scheduler decides the service rate. For example, the service rate of a higher priority queue is higher than that of a lower priority queue. The service rate of a queue can be specified by setting the scheduler properties. The scheduler may process the queues in priority order or based on a weighted round-robin schedule [13]. Let q_j^i denote the queue at the j^{th} priority level at switch n_i. At time slot t, each queue q_j^i has a packet arrival rate $\lambda_{i,j}(t)$ and a packet service rate $\mu_{i,j}(t)$. Therefore, the traffic intensity at q_j^i is $\rho_{i,j}(t) = \frac{\lambda_{i,j}(t)}{\mu_{i,j}(t)}$. For simplicity of the analysis, for each queue, we consider the $M/M/1/K$ queuing model where K is the finite queue length. In the $M/M/1/K$ queuing model, the incoming packets follow Poisson's distribution, and the service times of the packets are exponentially distributed [19]. The average queuing delay of a packet assigned to q_j^i is given by:

$$D_{i,j}(t) = \frac{1}{\mu_{i,j}(t)} \left(\frac{1 + K\rho_{i,j}(t)^{(K+1)} - (K+1)\rho_{i,j}(t)^K}{(1-\rho_{i,j}(t))(1-\rho_{i,j}(t)^{(K+1)})} \right) \quad (1)$$

Therefore, if a additional packets are assigned to q_j^i, the packet arrival rate for the time-slot t becomes $\lambda_{i,j}(t) + a$ and the traffic intensity is updated as $\rho'_{i,j}(t) = \frac{\lambda_{i,j}(t)+a}{\mu_{i,j}(t)}$.

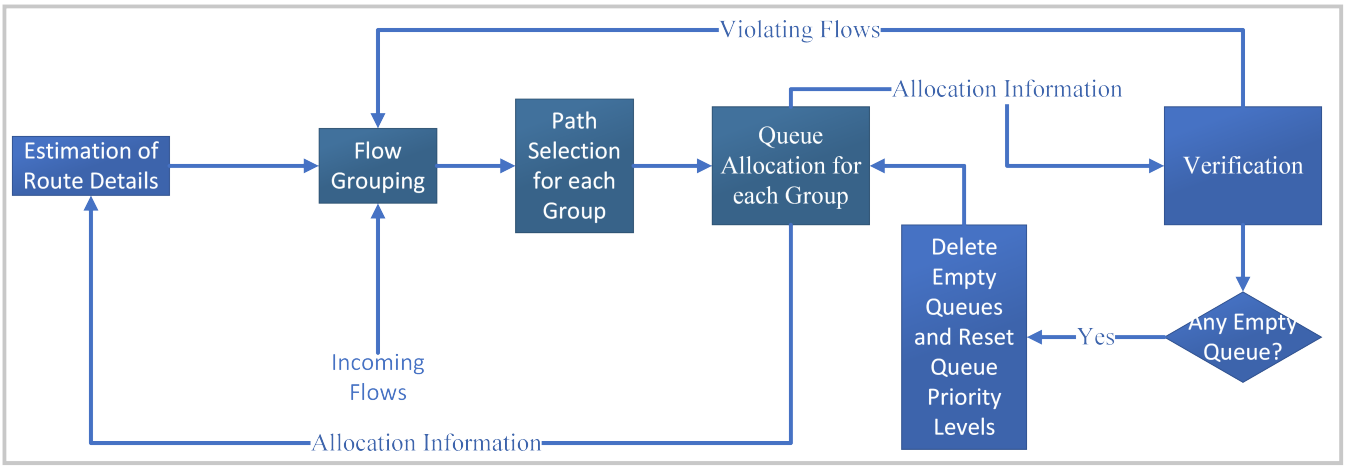


Fig. 2: Workflow of the proposed scheme

Accordingly, the increase in queuing delay is given by:

$$D^{'}_{i,j}(t) = \frac{1}{\mu_{i,j}(t)}\left(\frac{1+K(\rho^{'}_{i,j}(t))^{(K+1)}-(K+1)(\rho^{'}_{i,j}(t))^{K}}{(1-\rho^{'}_{i,j}(t))(1-(\rho^{'}_{i,j}(t))^{(K+1)})}\right) - D_{i,j}(t) \quad (2)$$

We assume that the number of queues per switch, the present arrival rate of each queue, the service rate of each queue, and the priority level of each queue are known.

B. Traffic Model

Let $F(t)$ denote the set of traffic flows and $P_x(t)$ refers to the set of packets belonging to a flow $f_x \in F(t)$ at time slot t. A flow $f_x \in F(t)$ is characterized by the maximum allowable delay (D_x^{max}), ordered set of switches in the flow path at time slot t ($path_x(t)$), source (src_x), and destination ($dest_x$). We define the following binary variable to denote the queue allocation for a flow f_x at time slot t:

$$\gamma_{x,i,j}(t) = \begin{cases} 1 & \text{if } f_x \text{ is allocated to } q^i_j \text{ at time slot } t, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Let δ^k_x denote the propagation delay between the k^{th} and $k+1^{th}$ switches in the set $path_x(t)$. Therefore, considering all packets at time slot t, the E2E latency of a flow f_x is:

$$L_x(t) = \max_{p_{x,y} \in P_x(t)} \Big(\sum_{n_i \in path_x(t)} \sum_{j=1}^{m^i} \gamma_{x,i,j}(t) D_{i,j}(t) + \sum_{k=1}^{|path_x(t)-1|} \delta^k_x \Big) \quad (4)$$

To differentiate traffic types, we assign a priority value to each flow. We get the priority value by normalizing the maximum allowable delay for the flow. After normalization, the priority value is approximated to the nearest integer. Therefore, for a flow f_x, the priority value is estimated as:

$$\sigma_x = \alpha + \frac{(D^{max}_x - min(D^{max}))\,(\beta-\alpha)}{max(D^{max}) - min(D^{max})}, \quad (5)$$

where $min(D^{max})$ and $max(D^{max})$ are the minimum and the maximum value of the maximum allowable delay. The normalization range is $[\alpha, \beta]$ with $\alpha > 0$. Here, low σ_x value signifies that f_x is a high priority flow. Let $B(t)$ denote the set of priority groups at time slot t, $F^g(t)$ denote the set of flows belonging to a group $b_g \in B(t)$ at time slot t, and σ^g be the priority value of group b_g. Therefore, if a flow f_x belongs to a priority group b_g, we get $\sigma_x = \sigma^g$ as all flows belonging to the same priority group has the same priority as estimated in Equation (5).

C. SDQ Framework

Software-defined queueing framework [6] installed on top of the control plane performs queue management and traffic engineering. Queue management involves the addition or deletion of queues, setting service rates, and scheduling algorithms for processing the queues. On the other hand, traffic engineering involves the selection of communication paths for incoming flows.

D. Problem Formulation

Given a set of switches N with each switch $n_i \in N$ having m_i queues associated with its output port. The goal of this work is to determine the path and queue allocation for the flows so that the E2E latency of each flow is less than the maximum allowable delay. Mathematically,

$$\underset{path(t),\gamma(t)}{\text{Minimize}} \sum_{b_g \in B(t)} \frac{1}{\sigma^g} \left(\sum_{f_x \in F^g(t)} \frac{L_x(t)}{D^{max}_x} \right) \quad (6)$$

subject to

$$L_x(t) \leq D^{max}_x, \forall f_x \in F(t) \quad (7)$$

IV. ReSQ: The Proposed Scheme

In this section, we present the proposed scheme ReSQ for latency-aware allocation of queues at each switch for the traffic flows.

A. Workflow of the Proposed Scheme

Fig. 2 shows the basic workflow of the proposed scheme. This process is triggered at the beginning of each time slot. Here, the concerned flows are new flows and the flows that violate E2E latency constraints based on the path selection and queue allocation in the previous time slot. The concerned flows are classified into multiple priority groups based on source, destination, and the maximum allowable delay. Starting with the highest priority group, the available shortest path is allocated to each group. Subsequently, an RL algorithm allocates queues to each group. Finally, we determine the violating flows and delete empty queues.

B. Flow Grouping

Existing traffic classification schemes can be used to classify the concerned flows in multiple priority groups [14] [15]. Similar to the work by Daly *et al.* [14], we use a hash function to classify the concerned flows based on respective priority. The hash key for a flow f_x is the tuple $key_x =< src_x, dest_x, \sigma_x >$. All packets of f_x are placed in a group with a key value the same as key_x. The process of assigning a flow to a priority group takes $O(|B(t)|)$ time in the worst case [14].

C. Path Selection

In this work, we assume that the system stores available paths between each pair of nodes, the hop count of each path, and the available bandwidth of each path [18]. For each priority group, the path is assigned. The path selection process for a priority group b_g is described in Algorithm 1. Algorithm 1 estimates the cumulative bandwidth demand for all flows in a priority group. The available paths are investigated in increasing order of hop count to satisfy the demands of URLLC traffic. If a path satisfies the required bandwidth demand, it is selected for the corresponding priority group. The path selection process is optimized by prioritizing the groups in priority order starting with the group with the highest priority. The selected paths and queues for existing non-violating flows remain unchanged. The time complexity of Algorithm 1 depends on the estimation of the total bandwidth demand of all flows in the respective priority group and the verification of available paths for a suitable selection. The first step takes $O(|F^g(t)|)$ time and the second step takes $O(path^g_{max})$ time where $path^g_{max}$ is the maximum number of paths between src and $dest$ nodes for the priority group b_g with key $\{src, dest, \sigma^g\}$. Therefore, the time complexity of Algorithm 1 is $O(|F^g(t)| + path^g_{max})$.

D. Queue Allocation

In 5G networks, the traffic matrix of changes frequently and the priority group formations change accordingly. Therefore, for the varying network dynamics, manually designing heuristic algorithms is tedious and inefficient. Q-learning (QL) is a widely used algorithm that requires no prior system knowledge and the system gradually learns to make better choices. QL algorithms converge with the maximum reward. Therefore, we design a Q-Learning (QL) framework to allocate queues for each priority group. We use QL to learn an optimal queue allocation policy on-demand for the new and existing flows without violating the respective latency constraints. Eventually, the QL-based system can satisfy the E2E latency constraint of the new and existing flows with minimum queue reallocation. Moreover, we speed up the learning process by using federated agents for each priority group sharing respective QL model weights.

Algorithm 1 Path Selection Algorithm (PSA) for b_g

INPUTS: Key for b_g ($key^g = \{src, dest, \sigma^g\}$), $F^g(t)$
OUTPUT: Selected path for b_g ($path^g(t)$)
PROCEDURE:
1: Estimate total bandwidth demand bw of the priority group
2: **for each** path between src and $dest$ **do**
3: $path^g(t) \leftarrow$ The unvisited shortest path
4: **if** $path^g(t)$ has residual bandwidth $bw^{res}_g(t)$ greater than bw **then**
5: $bw^{res}_g(t) \leftarrow bw^{res}_g(t) - bw$
6: **return** $path^g(t)$
7: **else**
8: $path^g(t) \leftarrow$ The next shortest path
9: **end if**
10: **end for**

For the RL algorithm, we define a Markov decision process as $< S, A, T, R, d >$, where S is the state space, A is the action space, $Pr(s_{\tau+1}|s_\tau a_\tau) \in T$ are the transition probabilities, $r_\tau(s_\tau a_\tau) \in R$ is the reward function, and $d \in [0, 1]$ is the discount factor [16]. The agent takes action based on the current state of the environment and the reward received for each action. The state of the environment changes after each action based on the respective transition probability. A policy $\pi(s, a) = Pr(a_\tau = 1|s_\tau = s)$ signifies the probability of taking action $a = a_\tau$ in state s_τ. The return at time τ is expressed as $R_\tau = \sum_{i=0}^{\infty} d^{i+\tau} r_{i+\tau}$. The goal of the agent is to determine an optimal policy $\pi^* = \underset{\pi}{argmax} E_\pi\{R_0|s_0 = s\}$. For a group b_g, we define state space, action space, and reward function as follows:

1) State Space

We define the state of the environment based on the traffic intensity of all queues in the communication path selected for the priority group. Let N^g be the set of switches in the selected path for priority group b_g. The state space of the corresponding environment is defined as:

$$S = \{s^j(\tau) = \{y^i_j(\tau)\}\}, \forall j \in [1, m_i], \forall n_i \in N^g, \quad (8)$$

$$y^i_j(\tau) = \begin{cases} 0, & \text{if } \rho_{i,j}(\tau) \leq 0.5, \\ 1, & \text{Otherwise}, \end{cases} \quad (9)$$

where $j \in [1, m_i]$ refers to the priority level of a queue at switch n_i and m_i is the total number of queues or priority levels at n_i. We consider an additional empty queue for each

switch to address the requirement of new queue formation. However, we assume that the maximum number of queues per switch is limited to M.

2) Action Space

An action signifies selecting a queue for a priority group. The action space for a queue is defined as:

$$A = \{a^j(\tau) = \{0, 1\}\}, \forall j \in [1, m_i], \forall n_i \in N^g, \quad (10)$$

where $a^j(\tau) = 0$ denotes that the queue is not allocated for the priority group and $a^j(\tau) = 1$ denotes that the queue is allocated for the priority group. Therefore, an action signifies the set of selected queues in the selected path for a priority group.

3) Reward Function

The goal of the agent is to minimize the number of flows that violate the E2E latency constraint after each action. Additionally, the agent aims to reduce the number of changes in the existing queue allocation. For an action $a(t)$, the value of reward $r(t)$ is defined as:

$$r_\tau = \frac{1}{\sigma^g} \left(\sum_{f_x \in F^g(t)} \frac{D_x^{max} - L_x(\tau)}{D_x^{max}(1 + v(\tau))(1 + c(\tau))} \right), \quad (11)$$

where $c(\tau)$ denotes the number of changes in the existing queue allocation for the current priority group, and $v(\tau)$ denotes the number of violating flows for the current priority group.

Algorithm 2 shows the steps of the queue allocation process based on QL. We consider the maximum number of episodes as Γ and the RL agent computes $\gamma(t)$. The parameters for the Q-learning algorithm are — learning rate α, discount factor z, and exploration policy ϵ. The time taken for queue allocation depends on the number of episodes, episode horizon time, and the size of the Q-table. Based on the formulated Markov decision process, the size of both the state space and the action space is $2^{|N^g|M}$ each. In reality, M is a user-defined parameter, and the value of $|N^g|$ is low as Algorithm 1 prioritizes shortest paths for path selection. Therefore, the time complexity of Algorithm 2 is $O(\Gamma H 2^{2|N^g|M})$.

Algorithm 2 Queue Allocation Algorithm (QAA)

INPUTS: Number of episodes (Γ), episode horizon time (H), Learning rate (α), exploration policy (ϵ), discount factor (z)
OUTPUT: $\gamma(t)$ ▷ queue allocation
PROCEDURE:
$Q(s_t, a_t) \leftarrow 0$, $\forall s_t \in S, \forall a_t \in A$ ▷ Q-table
for each $e \in \Gamma$ **do**
  for each $\tau \in H$ **do**
    Select $num \in [0, 1]$ randomly
    if $num > \epsilon$ **then**
      $a_\tau \leftarrow \underset{a}{\text{argmax}}\ Q(s_\tau, a)$
    else
      $a_\tau \leftarrow$ randomly selected action
    end if
    Take action a_τ and go to state $s_{\tau+1}$ to get reward $r_{\tau+1}$
    Update $\gamma(\tau)$
    $Q(s_\tau, a_\tau) \leftarrow (1 - \alpha)Q(s_\tau, a_\tau) + \alpha[r_\tau + z \max_a Q(s_{\tau+1}, a)]$ ▷ Q-table update
  end for
end for
return $\gamma(t)$

E. Verification

After completion of each time slot, the system identifies the flows that violate the E2E latency constraint. The path and queue allocation for these flows remains unchanged until the start of the next time slot. In the next time slot, flow grouping is performed for the violating flows as well as the new flows. The verification module also updates the current queue settings. If any queue is empty, it is deleted and the queue priority levels are reset. The identification of violating flows takes $O(|F(t)|)$ time. The identification of empty queues takes $O(|N|M)$ time in the worst case. Therefore, the worst-case time complexity of the verification module is $O(|F(t)| + |N|M)$.

V. Performance Evaluation

A. Simulation Settings

We use a discrete event simulator with a time slot duration of 50 ms to evaluate the performance of ReSQ. For the simulation, we consider the ATT topology having 25 switches [17] as the default topology. Additionally, we evaluate the performance of ReSQ with Abilene topology [17] which is a smaller topology than ATT. Table II shows the simulation parameters. The packet generation rate is varied between 200 packets/ms and 1000 packets/second for each time slot. Each flow has a fixed bandwidth demand and it varies between $0.003 - 5$ Mbps. The ϵ-greedy Q-learning agent considers the learning rate (α) as 0.1, discount factor (z) as 0.95 and

TABLE II: Simulation parameters

Parameter	Value
Topology	ATT, Abilene [17]
Number of Switches	25 (ATT), 11 (Abilene)
Maximum number of queues per switch	3
Packet generation rate	200-1000 packets/second
Bandwidth demand of a flow	$0.003 - 5$ Mbps
Maximum allowable delay	$5 - 300$ ms
Duration of a time slot	50 ms
Learning rate	0.1 [7]
Discount factor	0.95 [7]
Exploration policy	0.9 [7]

TABLE III: Traffic categories

Latency-sensitivity	Maximum allowable delay
High	$5 - 100$ ms
Medium	$101 - 200$ ms
Low	$201 - 300$ ms

the exploration policy (ϵ) as 0.9 [7]. For the simulation, we categorize the incoming flows into three traffic categories based on the latency-sensitivity as shown in Table III. Each experiment is conducted for 30 iterations and the 95% confidence interval is considered as the simulation results.

B. Benchmark Scheme

We consider random queue assignment (RQA) as a benchmark for the evaluation of the performance of ReSQ. RQA assigns the flows randomly to the queues in the flow path. Moreover, this scheme does not include a verification module to rectify the queue assignments of the violating traffic. For a fair comparison, we consider the same path selection procedure for both RQA and ReSQ. To evaluate the efficacy of the proposed path selection module, we compare ReSQ with greedy path selection (GPS) where the first available path with sufficient bandwidth is selected for each priority group. Additionally, we compare ReSQ with the initial SDQ solution proposed in [6] to show the necessity of the extension proposed in this work.

C. Performance Metrics

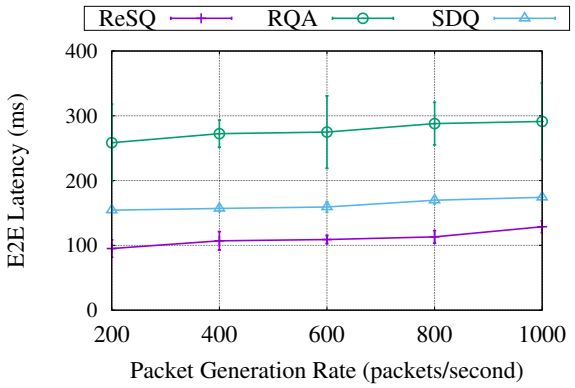
We consider the following metrics to analyze the performance of the proposed scheme:

- **E2E latency**: The E2E latency of each flow depends on the selected queue as mentioned in Equation (4).
- **QoS violation ratio**: QoS violation ratio expresses the number of flows having E2E latency higher than the corresponding maximum allowable delay values.
- **Flow path length**: This metric specifies the average hop count for the flows to evaluate the performance of the proposed path selection module.

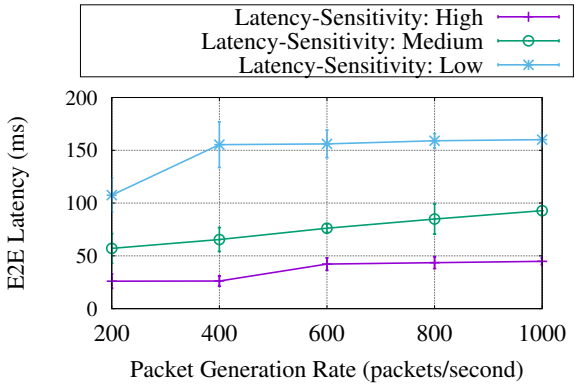
D. Result and Discussion

1) E2E Latency

From As a performance metric, we analyze the E2E latency of flow which includes the propagation delay and the queueing delay at each switch in the respective routing path as expressed in Equation (4). This metric is one of the important performance measures, especially for URLLC traffic having strict latency demands. Fig. 3a, we observe that the average E2E latency of ReSQ is 60.07% less than that of RQA. This is because ReSQ allocates queues more optimally considering specific QoS demands. In particular, for ReSQ, the RL agent selects an action or queue allocation that reduces the aggregated E2E latency for a priority group. In contrast, random queue allocation by RQA increases queueing delay of some queues and results in high average E2E latency. Additionally, we observe that the average E2E latency of ReSQ is 32.1% less than that of SDQ. This is because SDQ selects a less loaded path to route the traffic and does not consider the length of the path which influences the propagation delay. However, ReSQ selects the shortest path that has enough bandwidth to accommodate the request and



(a) Comparison with RQA



(b) Performance of ReSQ for different traffic categories

Fig. 3: ATT Topology: E2E latency

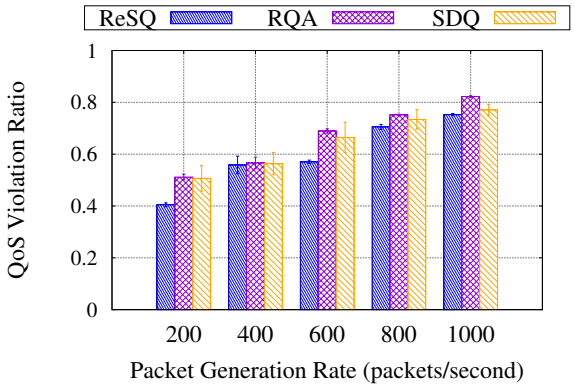
the RL agent minimizes the E2E delay further by selecting appropriate queues with low delay.

In a network having heterogeneous traffic with different latency demands, it is important to serve the highly latency-sensitive traffic earlier than other traffic. Therefore, we compare the performance of ReSQ for different traffic classes categorized by respective latency demands. From Fig. 3b, we observe that highly latency-sensitive traffic experiences significantly low E2E delay as compared to other traffic categories. This is because ReSQ selects the set of optimal queues that reduces the gap between the maximum allowable latency and the E2E latency as stated in Equation (11). For example, a highly latency-sensitive flow f_x has a low maximum allowable delay D_x^{max}. For this flow, the RL agent selects queues so that the E2E latency $L_x(\tau)$ is lower and the difference $(D_x^{max} - L_x(\tau))$ is higher to obtain a higher reward. Therefore, E2E latency decreases with increasing latency-sensitivity of the flows.

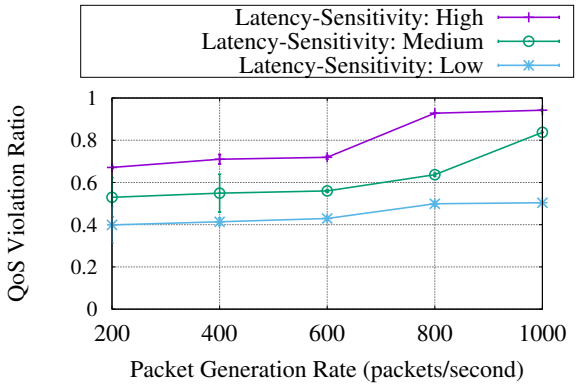
2) QoS Violation Ratio

We consider the QoS violation ratio as a performance metric to measure the number of flows that fail to complete within the respective delay budget due to inefficient path selection and queue allocation. Fig. 4a shows that the QoS violation ratio of ReSQ is 10.45% less than that of RQA. This is because of cautious queue allocation for latency-sensitive flows. In particular, the RL agent aims to minimize

the number of violating flows to achieve a higher reward as mentioned in Equation (11). Moreover, each RL agent periodically rectifies its earlier decision in subsequent time slots and modifies queue allocation for violating traffic. The average QoS violation ratio of ReSQ is 7.63% less than that of SDQ. This is because SDQ does not consider the reallocation of violating traffic. Whereas, ReSQ performs flow grouping, path selection, and queue allocation for violating traffic to reduce further QoS violations.



(a) Comparison with RQA



(b) Performance of ReSQ for different traffic categories

Fig. 4: ATT Topology: QoS violation ratio

We analyze the QoS violation of ReSQ for different traffic categories. From Fig. 4b, we observe that the QoS violation ratio for highly latency-sensitive traffic is high. This is because the maximum allowable delay values of these flows are very low as shown in Table III.

3) Flow Path Length

Low flow path length implies low aggregated propagation delay which in turn reduces the E2E latency as stated in Equation (4). Therefore, we compare the average flow path length achieved by ReSQ with that of the benchmarks. Fig. 5 shows that the average flow path length for ReSQ is significantly lower than that of GPS. This is because Algorithm 1 explores the paths in the order of hop count and selects the path with sufficient capacity. On the other hand, the GPS method greedily selects the first available path with sufficient bandwidth. Therefore, in this case, a selected path may have a larger length. From Fig. 5, we also observe that SDQ has

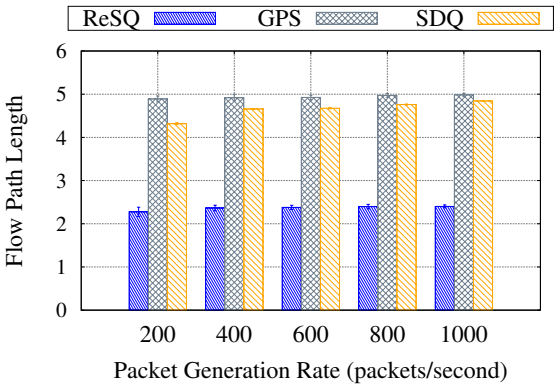
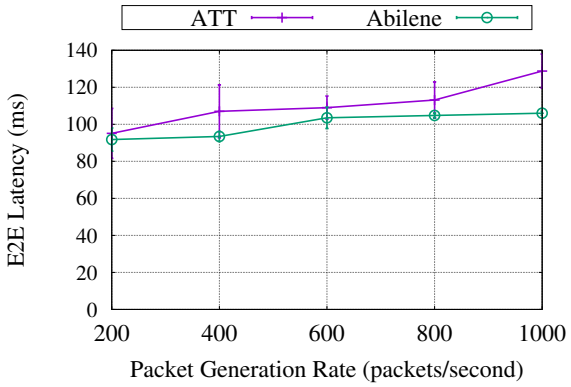
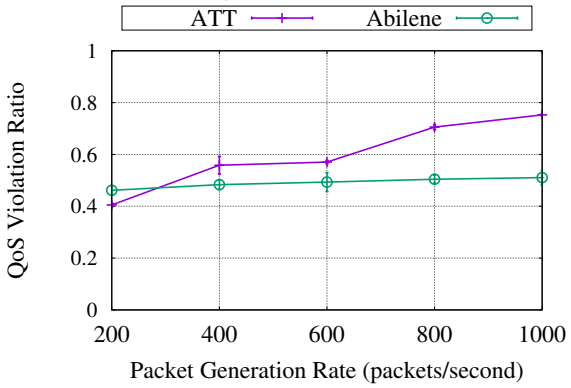


Fig. 5: ATT Topology: Flow Path Length

a significantly higher path length compared to ReSQ. This is because SDQ prioritizes available paths based on the available bandwidth in each path. SDQ selects the least loaded path. Therefore, in SDQ, achieving the minimum path length is not guaranteed.



(a) E2E latency



(b) QoS violation ratio

Fig. 6: Performance of ReSQ for ATT and Abilene topologies

Topology size has a significant effect on path and queue selection. A large-scale topology has multiple options for path selection which result in some paths with high propagation delay because of high hop count. On the other hand, the selections of path and queue are limited in smaller topologies due to less number of available options. Therefore, it is

interesting to observe the performance of ReSQ for topologies having different dimensions. Fig. 6 evaluates the performance of ReSQ for ATT and Abilene topologies. From Fig. 6a, we observe that E2E latency for Abilene is less than that for ATT. This is because Abilene is a smaller topology than ATT and the number of switches in each flow path is low resulting in low propagation delay. For the same reason, the average QoS violation ratio for the Abilene topology is low. However, as shown in Fig. 6b, for low packet generation rates such as 0.2 packets/second, the QoS violation for Abilene is higher than that of ATT. This is because ATT topology has more paths available to well-distribute the traffic so that the queuing delay is less at each switch in the flow path. However, as the traffic load increases the increase in propagation delay increase the E2E latency, and the QoS violation ratio increases as well for the ATT topology.

VI. Conclusion

This paper extends the concepts of the software-defined queueing framework by automating the selection of routing paths and queues for traffic flows belonging to heterogeneous priority classes. The proposed scheme ReSQ employs a hashing-based technique to classify incoming packets into multiple priority groups. Additionally, QL is used to assign queues to the priority groups in parallel. One of the strengths of ReSQ is that it automates the queue allocation and queue setting update in the SDQ framework. Additionally, ReSQ routes flow through optimally selected paths with low hop counts to reduce the propagation delay which is a significant component of E2E latency. Accordingly, the performance evaluation shows promising results in terms of E2E latency and QoS violation ratio.

ReSQ re-allocates path and queue for all violating flows which may not be feasible because there may be some violating flows that can never fulfill the respective latency demands. Therefore, in the future, we plan to extend this work by selecting violating flows for which path and queue re-allocation is feasible. Additionally, we plan to consider multiple output ports associated with each switch.

Acknowledgment

This work is supported by the CHARITY project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016509. It is also supported in part by the Academy of Finland 6Genesis project under Grant No. 318927 and IDEA-MILL with grant number 33593.

References

[1] J. Prados-Garzon, T. Taleb, and M. Bagaa, "LEARNET: Reinforcement Learning Based Flow Scheduling for Asynchronous Deterministic Networks," in *IEEE ICC*, 2020, pp. 1–6.

[2] M. Khoshnevisan, V. Joseph, P. Gupta, F. Meshkati, R. Prakash, and P. Tinnakornsrisuphap, "5G Industrial Networks With CoMP for URLLC and Time Sensitive Network Architecture," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 947–959, 2019.

[3] A. Montazerolghaem and M. H. Yaghmaee, "Load-Balanced and QoS-Aware Software-Defined Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3323–3337, 2020.

[4] J. W. Guck, A. Van Bemten, and W. Kellerer, "DetServ: Network Models for Real-Time QoS Provisioning in SDN-Based Industrial Environments," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1003–1017, 2017.

[5] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring End-to-End QoS Based on Multi-Paths Routing Using SDN Technology," in *IEEE GLOBECOM*, 2017, pp. 1–6.

[6] 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.

[7] D. Ghosal, S. Shukla, A. Sim, A. V. Thakur, and K. Wu, "A Reinforcement Learning Based Network Scheduler for Deadline-Driven Data Transfers," in *IEEE GLOBECOM*, 2019, pp. 1–6.

[8] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic Engineering With Reinforcement Learning in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, 2020.

[9] X. Guo, H. Lin, Z. Li, and M. Peng, "Deep-Reinforcement-Learning-Based QoS-Aware Secure Routing for SDN-IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6242–6251, 2020.

[10] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. S. da Fonseca, "Intelligent Routing Based on Reinforcement Learning for Software-Defined Networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 870–881, 2021.

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

[12] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *Telecommunications Forum (TELFOR)*, 2014, pp. 111–114.

[13] inSpace, "Queuing Mechanism in Modern Switches," Tech. Rep., Accessed on: February 24, 2021. [Online]. Available: https://blog.ipspace.net/2014/05/queuing-mechanisms-in-modern-switches.html

[14] J. Daly, V. Bruschi, L. Linguaglossa, S. Pontarelli, D. Rossi, J. Tollet, E. Torng, and A. Yourtchenko, "TupleMerge: Fast Software Packet Processing for Online Packet Classification," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1417–1431, 2019.

[15] W. Li, T. Yang, O. Rottenstreich, X. Li, G. Xie, H. Li, B. Vamanan, D. Li, and H. Lin, "Tuple Space Assisted Packet Classification With High Performance on Both Search and Update," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1555–1569, 2020.

[16] P. Tehrani, F. Restuccia, and M. Levorato, "Federated Deep Reinforcement Learning for the Distributed Control of NextG Wireless Networks," in *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2021, pp. 248–253.

[17] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[18] I. Maity, S. Misra, and C. Mandal, "ETHoS: Energy-Aware Traffic Engineering for Sustainable Hybrid SDN," *IEEE Transactions on Sustainable Computing*, doi: 10.1109/TSUSC.2022.3164571.

[19] I. Maity, A. Mondal, S. Misra, and C. Mandal, "CURE: Consistent Update with Redundancy Reduction in SDN," *IEEE Transactions on Communications*, vol. 66, no. 9, pp. 3974–3981, Sep. 2018.

[20] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[21] A. Mondal, S. Misra, and I. Maity, "Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1359-1366, Jun. 2019.

[22] M. Rahouti, K. Xiong, Y. Xin and N. Ghani, "A Priority-Based Queueing Mechanism in Software-Defined Networking Environments," *IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2021, pp. 1-2.

[23] E. Li, F. He, L. Zhao and X. Zhou, "A SDN-based Traffic Bandwidth Allocation Method for Time Sensitive Networking in Avionics," *IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2019, pp. 1-7.

[24] Y. Goto, H. Masuyama, B. Ng, W. K. G. Seah and Y. Takahashi, "Queueing Analysis of Software Defined Network with Realistic OpenFlow–Based Switch Model," *IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2016, pp. 301-306.