Reinforcement Learning-based Traffic Engineering for QoS Provisioning and Load Balancing

Minghao Ye[†*], Yang Hu[‡*], Junjie Zhang[‡], Zehua Guo[§¶], H. Jonathan Chao[†]
[†]New York University [‡]Fortinet, Inc. [§]Beijing Institute of Technology
Email: {minghao.ye, yh3751, junjie.zhang, chao}@nyu.edu, guolizihao@hotmail.com

***Abstract*—Emerging applications pose different Quality of Service (QoS) requirements for the network, where Traffic Engineering (TE) plays an important role in QoS provisioning by carefully selecting routing paths and adjusting traffic split ratios on routing paths. To accommodate diverse QoS requirements of traffic flows under network dynamics, TE usually periodically computes an optimal routing strategy and updates a significant number of forwarding entries, which introduces considerable network operation management overhead. In this paper, we propose QoS-RL, a Reinforcement Learning (RL)-based TE solution for QoS provisioning and load balancing with low management overhead and service disruption during routing updates. Given the traffic matrices that represent the traffic demands of high and low priority flows, QoS-RL can intelligently select and update only a few destination-based forwarding entries to satisfy the QoS requirements of high priority traffic while maintaining good load balancing performance by rerouting a small portion of low priority traffic. Extensive simulation results on four real-world network topologies demonstrate that QoS-RL provides at least 95.5% of optimal end-to-end delay performance on average for high priority flows, and also achieves above 90% of optimal load balancing performance in most cases by updating only 10% of destination-based forwarding entries.**

***Index Terms*—Reinforcement Learning, Traffic Engineering, Quality of Service, Load Balancing, Management Overhead**

I. Introduction

In today's Wide Area Networks (WANs), network operators usually configure routing policies for different traffic flows to improve network performance with Traffic Engineering (TE) frameworks. Generally speaking, TE can properly distribute network traffic by optimizing the routing strategy with a performance objective [1]–[4]. For instance, Internet Service Providers (ISPs) can target minimizing the Maximum Link Utilization (MLU) in their networks to achieve good load balancing performance and reduce the congestion probability. Cloud service providers [5], [6], who own a private WAN to connect their data centers, could have different objectives with consideration of resource utilization and operational cost, such as maximizing throughput. To achieve the goal specified by network operators, TE needs to periodically collect a Traffic Matrix (TM) that represents the recently measured demand volume of different traffic flows. Based on the network topology information and measured TM, TE can compute an optimal routing strategy with an aim to improve network performance, and then update the routing in the underlying network devices to enforce new routing policies to accommodate varying traffic demands.

Nowadays, emerging applications pose different Quality of Service (QoS) requirements on WANs. For instance, online gaming, metaverse, and video conferencing are usually delay-sensitive since a high latency would greatly impair users' quality of experience. Data analytics and batch computing applications could be delay-tolerant while requesting sufficient network bandwidth for data transmission. In this situation, it is critical for TE to satisfy diverse QoS requirements of different applications while achieving the performance objective of network operators. Fortunately, Software-Defined Networking (SDN) [7] is an efficient solution for TE to improve network performance and provide good QoS simultaneously. With a global view of the network, SDN-based TE can adaptively generate per-flow routing policies for different applications at the controller, and then dynamically deploys them at the underlying SDN switches to realize fine-grained traffic control.

However, enabling flexible and fine-grained routing for each 5-tuple flow may negatively impact the overall network operations due to considerable management overhead. First, the routing update process of the per-flow routing could saturate the processing capacity of the controller with high computation and communication overhead. Considering various QoS requirements of different applications, it would be time-consuming for TE to compute an optimal routing strategy for each flow in large networks [8]. Moreover, TE needs to generate and deploy multiple flow-based forwarding entries for each flow at different switches to establish or update its forwarding path. Such a large number of forwarding entry updates could overwhelm the controller with path calculation and forwarding entry generation/deployment [9]. Second, fine-grained routing policies could pose a great challenge to the storage capacity of SDN switches. Generally, SDN switches use Ternary Content-Addressable Memory (TCAM) to construct their flow tables to support flexible flow matching [10]. However, the TCAM resources in SDN switches are limited due to the high cost and power consumption [11]–[13]. As a result, SDN switches cannot accommodate a large number of per-flow forwarding entries [10], [14].

To reduce the adversary impact of flow-based routing updates as mentioned above, we propose QoS-RL, a Reinforcement Learning (RL)-based TE solution that selectively updates a few destination-based forwarding entries to maintain good QoS and improve load balancing performance simultaneously. Instead of using per-flow entries to match the 5-tuple infor-

* Co-first authors.
¶ Corresponding author.

979-8-3503-9973-8/23/$31.00 ©2023 IEEE

mation in packet headers, QoS-RL employs destination-based entries that only match the destination IP address of traffic flows, which greatly reduces the number of entries and overall management overhead. To distinguish the traffic of different applications with diverse QoS requirements under destination-based routing, we can categorize traffic into different priority levels at the sender side by tagging the packet headers [6], [15]. For example, the traffic of delay-sensitive applications can be tagged as high priority traffic, while the remaining delay-tolerant traffic is viewed as low priority traffic [16]. In this situation, the measured network TM can be decomposed into a high priority TM for QoS provisioning purposes and a low priority TM for achieving load balancing objectives.

To further reduce network management overhead and service disruption, QoS-RL adopts a two-step optimization approach with Linear Programming (LP) coupled with intelligent decisions from RL agents. By default, all flows are routed by static forwarding rules, such as Equal-Cost Multipath (ECMP) [17]. To accommodate dynamic network conditions, we leverage RL to determine which entries should be updated to reroute high/low priority traffic with respect to different objective functions (e.g., minimize delay or MLU), and then use LP to optimize the traffic split ratios for these entries among available next hops. When a high priority TM is measured, QoS-RL can intelligently insert a few high priority destination-based forwarding entries at selected SDN switches to reroute some high priority flows with an aim to provide good QoS (e.g., low latency). Once the routing decisions for high priority traffic are finalized, the allocated bandwidth should be reserved to guarantee QoS. Then, QoS-RL would identify another set of low priority forwarding entries based on the low priority TM to achieve good load balancing by rerouting a small portion of low priority traffic. Since low priority traffic is usually delay-tolerant, a preset performance objective of network operators (e.g., minimize MLU) can be considered as the target of LP to obtain optimal traffic split ratios for these low priority forwarding entries. As a result, QoS-RL can achieve good QoS provisioning and promising load balancing performance by only updating 10% of total forwarding entries with considerable flow entry savings.

The main contributions of this paper are summarized as follows:

1) We propose a TE solution called QoS-RL that achieves promising load balancing performance for network operators and provides good QoS for users simultaneously.
2) We leverage the agility of emerging RL to intelligently update a few destination-based forwarding entries for different priorities of traffic to reduce management overhead and mitigate service disruption.
3) Evaluation results on real-world WAN topologies show that QoS-RL can provide at least 95.5% of optimal delay on average for high priority traffic while achieving above 90% of optimal load balancing performance in most cases. Besides, QoS-RL generalizes well to unseen traffic scenarios with low overhead/disturbance by only updating 10% of destination-based forwarding entries.

The remainder of this paper is organized as follows. Section II presents the problem statement and explains our motivation. Section III provides the system description of QoS-RL. Section IV describes our RL design to identify important destination-based forwarding entries for different priorities of traffic. Section V describes how to calculate traffic split ratios of the selected entries with different optimization targets. Section VI evaluates the effectiveness of our scheme. Section VII lists the related works, and Section VIII concludes the paper.

II. Background and Motivation

A. Problem Statement

Generally, a network can be modeled as a directed graph $G(V, E)$, where V is the collection of network nodes (i.e., routers/switches) and E is the collection of directed links. Each directed link $\langle i, j\rangle \in E$ has a capacity $c_{i,j}$, which specifies the maximum available bandwidth that can be allocated to traffic flows. If the link load $l_{i,j}$ is too large, the congestion probability would increase and eventually lead to packet loss. Given the recently measured traffic demands from all source nodes $s \in V$ to all destination nodes $d \in V$, a high priority TM and a low priority TM can be constructed to represent the average demand volume of all source-destination flows $\langle s, d\rangle$ with different priorities in the past time interval, respectively. To efficiently deliver high priority traffic and low priority traffic from their sources to destinations, it is important to properly configure routing strategies based on traffic demands and available resources. Otherwise, the network may experience severe performance degradation with high latency and potential packet loss under bad routing decisions.

To provide good QoS to users and maintain promising network performance under dynamic traffic scenarios, network operators usually adopt TE frameworks to optimize the routing strategy periodically based on recently measured TMs. Given the network topology information and high/low priority TMs, TE can formulate and solve an optimization problem with different performance objectives to obtain optimal traffic allocations. To satisfy the delay requirements of high priority traffic, TE can target minimizing the overall end-to-end delay with the high priority TM. As described in [4], the objective function can be formulated as $\min \sum_{\langle i,j\rangle \in E} l_{i,j}/(c_{i,j} - l_{i,j})$ according to the M/M/1 queuing system, which aims at minimizing the average network delay experienced by packets [18]. For the low priority TM, TE can focus on achieving the network performance objective specified by network operators since low priority traffic is usually delay-tolerant. A common TE objective is minimizing the MLU in the network to achieve good load balancing. Here, MLU denotes the utilization of the most congested link in the network, and the objective function can be formulated as $\min \max_{\langle i,j\rangle \in E}(l_{i,j}/c_{i,j})$ [18].

An effective approach to guarantee good service quality is performing multi-step TE optimization to allocate traffic flows in descending order of priority [6]. For example, TE can first optimize the routing for high priority delay-sensitive flows, such that high priority traffic would be properly distributed

among shorter paths to avoid congestion while experiencing lower latency. The allocated bandwidth for high priority traffic should be reserved at different links for good QoS provisioning, which cannot be reused by low priority traffic. After that, TE would allocate low priority traffic for achieving good load balancing performance. In other words, low priority traffic could be forwarded along longer paths with higher latency since these links are underutilized. Given that low priority flows usually do not have stringent delay requirements, it would be acceptable to focus on load balancing performance at this stage.

B. Motivation

As we discussed in Section I, fine-grained forwarding rules for 5-tuple flows would lead to high management overhead in the network, including the computation and communication overhead to generate and deploy a large number of flow entries as well as severe TCAM resource consumptions. For a network with N nodes and L different applications, there will be $O(N^3L)$ forwarding entries to be generated and deployed in total, considering $N \cdot (N-1)$ source-destination pairs, at most N switches to be updated, and L types of applications. To alleviate this issue, we adopt a relatively coarse-grained approach: (1) divide network traffic into different levels of priorities of TMs (e.g., two levels as high/low priority) based on the QoS requirements of L different applications, and (2) use destination-based forwarding entries to control traffic distribution and provide good QoS to high priority traffic while maintaining promising load balancing performance. As a result, only $O(N^2)$ entries are needed since each node's forwarding entry only matches with a destination address (assuming that the number of priority classes is a constant), and the overall management overhead can be greatly reduced.

Another important aspect is to mitigate potential service disruption and network disturbance caused by rerouting operations of TE. According to recent studies [19], [20], frequent routing updates could disrupt network services with packet reordering issues and result in reduced throughput, frequent latency variations, and longer flow completion time. Even though the overall management overhead is reduced with TM priority classification and destination-based forwarding, TE could reroute a large number of flows during routing updates with a severe impact on service quality. To mitigate the negative impact of rerouting operations, we propose to selectively update a few destination-based forwarding entries (e.g., 10% of total entries) instead of all entries, such that most flows are still routed on their previously assigned path without disruption. However, given the large solution space, it is very challenging to find a good set of forwarding entries for TE to maintain good load balancing performance and satisfy QoS requirements. As we verified in Section VI, simple rule-based heuristics cannot adapt to dynamic network conditions with sub-optimal load balancing performance and degraded QoS.

To identify a good set of forwarding entries to be updated, we leverage the agility of emerging RL techniques to learn a selection policy that can adapt to different traffic scenarios.

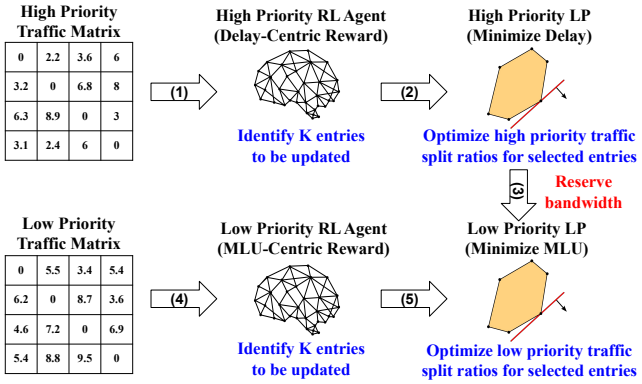


Fig. 1: Workflow of QoS-RL.

RL has been demonstrated to work well in addressing complex decision-making problems, as it can effectively extract hidden patterns from raw observations (e.g., TMs) and continuously learn from interacting with the environment and receiving reward signals to improve its strategy. A well-trained RL agent is capable of generalizing to different input states that are unseen before. Moreover, RL does not require labeled data for training but purely learns to reinforce the actions that lead to higher rewards, which is suitable for identifying a good set of forwarding entries. Since there is no "ground truth" regarding which entries are important for improving load balancing performance and QoS, we can design a reward function for RL to automatically learn a good selection policy that matches our objective. Once the most important forwarding entries are identified by the RL agent, it would be relatively easy to obtain the optimal traffic split ratios for the selected entries by solving an LP problem. Then, we can configure these entries at the underlying switches accordingly.

III. System Description

QoS-RL is an RL-based TE framework deployed in a controller to configure the routing policy in the network with the objective to achieve good QoS provisioning and load balancing. Fig. 1 illustrates the workflow of QoS-RL, which can be divided into five steps. Generally, the controller would periodically measure a high priority TM and a low priority TM representing the most recent traffic demands. Since QoS-RL allocates network traffic in priority order, the high priority TM is first taken as input to the high priority RL agent, which is trained with a delay-centric reward function to provide good QoS. Then, the RL agent will identify K destination-based forwarding entries that should be updated to reroute high priority traffic. Given the selected entries, QoS-RL solves an LP optimization problem (6) to obtain the optimal traffic split ratios for these entries to minimize the overall delay of high priority traffic. The allocated bandwidth on each link would be reserved such that it cannot be reused when allocating low priority traffic. The rest of the workflow for the low priority TM is similar to the first two steps. The only difference is that the low priority RL agent is trained with an MLU-centric reward function, and the objective of the low priority LP is to minimize the MLU to achieve good load balancing.

Fig. 2 provides an illustrative example of QoS-RL's operation to generate and update K forwarding entries. By default, all traffic is forwarded by static ECMP without considering traffic priorities and QoS/performance objectives. Since all traffic is forwarded on the shortest paths, the network becomes heavily congested. As depicted in Fig. 2(a), the two links $\langle 1, 4\rangle$ and $\langle 4, 9\rangle$ become bottleneck links with 90% and 100% utilization, respectively, which greatly impairs the overall end-to-end delay performance. In this situation, QoS-RL would consider different priorities of traffic and allocate high priority traffic first. From Fig. 2(b), we can see that switch 1 sends 3 units and 2 units of high priority traffic to switch 4 and switch 9, respectively, and switch 4 also sends 4 units of high priority traffic to switch 9. If all traffic is allocated to the shortest paths, high link utilization may increase the queuing delay of packets. Therefore, QoS-RL generates and updates a high priority destination-based forwarding entry at switch 1 to direct high priority traffic $t^H_{1,9}$ to switch 2. Once the traffic arrives at switch 2, it will be forwarded by default ECMP routing to reach switch 9, which results in a secondary shortest path forwarding to guarantee low latency. Once the high priority bandwidth allocation is finalized, QoS-RL would focus on the allocation of low priority traffic. Since low priority traffic is usually delay-tolerant, QoS-RL can allocate these flows to longer paths that are underutilized to achieve good load balancing. By inserting a low priority forwarding entry at switch 1, 4 units of low priority traffic in $t^L_{0,10}$ and $t^L_{1,10}$ would be directed to switch 3, as depicted in Fig. 2(c). Considering the traffic demands of $t^L_{5,8}$ and $t^L_{6,8}$, QoS-RL further splits the aggregated low priority traffic of $t^L_{0,10}$ and $t^L_{1,10}$ at switch 3 with different split ratios to minimize MLU. As a result, QoS-RL only inserts one high priority forwarding entry and two low priority forwarding entries to provide good QoS for high priority traffic and reduce the MLU by 60% with promising load balancing performance. Note that high priority entries should take strict precedence over low priority entries, and low priority entries take strict precedence over ECMP entries.

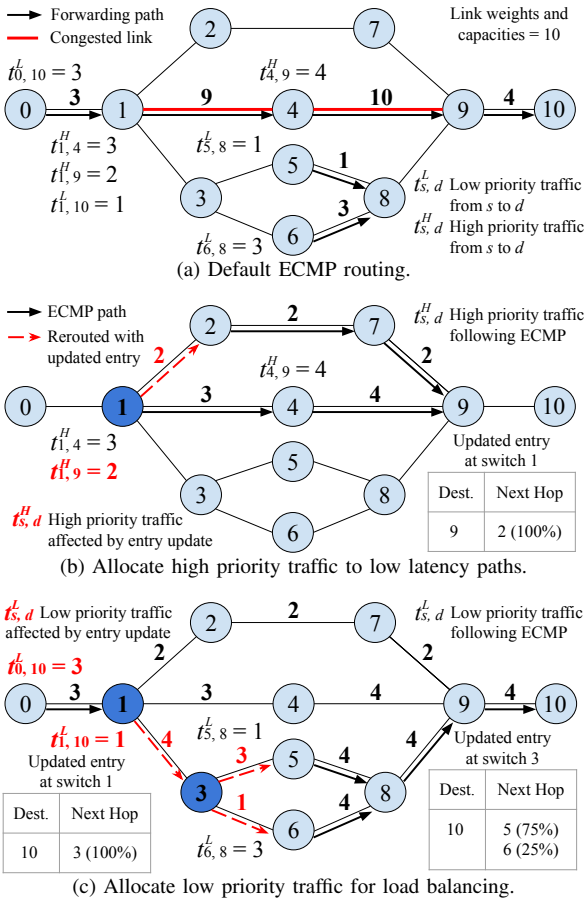


Fig. 2: Example of QoS-RL to reroute different priorities of traffic with several selected destination-based forwarding entries.

IV. Reinforcement Learning Formulation

In this section, we describe the RL design and algorithm for QoS-RL to learn a policy that selects a good combination of K destination-based forwarding entries for routing updates.

A. Agent Design

To properly handle traffic flows with different priority levels, we train a high priority RL agent and a low priority RL agent separately with different objective functions.

Input State: For different priorities of traffic, a TM is periodically measured with the most recent traffic demands in the past time interval, which can be further decomposed into a high priority TM and a low priority TM. To capture hidden traffic patterns and accommodate dynamic traffic scenarios, the high priority RL agent takes a high priority TM at time step t as an input (i.e., $s_t = TM^H_t$), while the low priority TM at time step t is observed by the low priority RL agent (i.e., $s_t = TM^L_t$).

Action Space: Given an input TM, the task of an RL agent is to identify a good set of K entries that contributes the most to improving QoS/load balancing performance. Under the context of destination-based forwarding, there are at most $N \cdot (N-1)$ candidate entries in a network with N nodes, considering N switches to be updated and $N-1$ destination nodes. Therefore, the action space can be formulated as $\{0, 1, ..., N \cdot (N-1) - 1\}$, where each element refers to a unique destination-based forwarding entry. An RL agent can sample K entries from the action space as a solution a_t for a given input s_t, which is denoted as $a_t = \{a^1_t, a^2_t, ..., a^K_t\}$.

Reward Function: For the high priority RL agent, the objective is to minimize the end-to-end delay with good QoS provisioning. Once K actions are sampled by the high priority RL agent based on an input $s_t = TM^H_t$, they can be passed to the LP formulation (6) to optimize routing with the objective to minimize the delay. The resulting end-to-end delay performance D is then fed into the high priority RL agent as part of the reward signal. Since the network delay may vary under different traffic scenarios and routing strategies, it is essential to compare D against the optimal delay D_{opt} that is obtained by updating all $N \cdot (N-1)$ entries with traffic split ratios optimized by LP for minimizing the end-to-end delay.

Thus, we define a delay performance ratio PR_D as follows:

$$PR_D = D_{opt}/D. \quad (1)$$

Since the optimal delay D_{opt} is always the lowest value, the delay performance ratio should be capped at 1. A higher PR_D value can be interpreted as better delay performance. Thus, the reward function of the high priority RL agent can be configured as $r_t = PR_D$ to reflect the resulting delay performance after updating the selected forwarding entries.

The low priority RL agent mainly focuses on achieving the performance target from the perspective of network operators, such as minimizing the MLU with good load balancing performance. Given an input low priority TM $s_t = TM_t^L$, the low priority RL agent samples K actions and then uses the LP formulation (8) to solve for optimal traffic split ratios in terms of MLU minimization. Similarly, the resulting MLU U is compared to the optimal MLU U_{opt} that is obtained by updating all $N \cdot (N-1)$ entries with traffic split ratios optimized by an LP, whose objective is to minimize the MLU. Then, we define an MLU performance ratio PR_U as follows:

$$PR_U = U_{opt}/U. \quad (2)$$

When PR_U = 1, QoS-RL achieves the optimal load balancing performance. The higher the PR_U, the better the network performance. Therefore, we take $r_t = PR_U$ as the reward function of the low priority RL agent to encourage better entry selections toward better load balancing performance.

B. Training Algorithm

For each of the RL agents, we use a neural network to represent the forwarding entry selection policy. Given a high/low priority TM as an input, the output of the neural network is a probability distribution $\pi(a_t|s_t)$ over the action space $\{0, 1, ..., N \cdot (N-1) - 1\}$. The goal of the training procedure of QoS-RL is to maximize the expected reward $E[r_t]$, i.e., maximizing QoS/load balancing performance over dynamic traffic scenarios. Thus, we use the REINFORCE algorithm [21] with a baseline $b(s_t)$ to optimize $E[r_t]$ by gradient ascent. Since each RL agent would sample K actions for each input state s_t, the parameters θ of the policy network should be updated as follows:

$$\theta \leftarrow \theta + \alpha \sum_t \sum_{i=1}^{K} \nabla_\theta \log \pi(a_t^i|s_t;\theta)(r_t - b(s_t)), \quad (3)$$

where α is the learning rate for the policy network. A good baseline $b(s_t)$ configuration can effectively reduce the gradient variance and facilitate the learning process. To calculate the baseline $b(s_t)$, we need an estimate of the value function $V^{\pi_\theta}(s_t)$, i.e., the expected accumulated reward starting at initial state s_t and following the policy π_θ. Thus, we use another neural network as the critic network for each RL agent, which is trained to learn an estimate of $V^{\pi_\theta}(s_t)$. The critic network parameter θ_v is updated as follows:

$$\theta_v \leftarrow \theta_v - \alpha_v \sum_t \nabla_{\theta_v} (r_t - v^{\pi_\theta}(s_t;\theta_v))^2, \quad (4)$$

where $v^{\pi_\theta}(\cdot;\theta_v)$ is the output of the critic network as the estimate of $V^{\pi_\theta}(\cdot)$, and α_v is the learning rate for the critic network. This critic network is only trained to estimate the expected accumulated reward starting at the initial state s_t and provide a good baseline to accelerate the training process of the policy network. In other words, the critic network is no longer required once the training process is completed. A well-trained RL agent can make decisions solely with the policy network during online deployment.

One important aspect to be considered during RL training is the trade-off between exploitation and exploration. If the RL agent focuses too much on exploitation, it could be trapped in a sub-optimal policy. Therefore, we need to properly encourage exploration during RL training to discover potential good policies for long-term benefits. As suggested by [22], we add the entropy of the policy π to Eq. (3) to improve the exploration of the RL agent:

$$\theta \leftarrow \theta + \alpha \sum_t (\sum_{i=1}^{K} \nabla_\theta \log \pi(a_t^i|s_t;\theta)(r_t - v^{\pi_\theta}(s_t;\theta_v)) + \beta \nabla_\theta H(\pi(\cdot|s_t;\theta))), \quad (5)$$

where H is the entropy of the policy (i.e., the probability distribution over actions), and the hyperparameter β is responsible for controlling the strength of the entropy regularization term.

C. Neural Network Architecture

We use TensorFlow [23] to implement the neural network architecture of QoS-RL. The policy neural network consists of three layers. The first layer is a convolutional layer with 128 filters, and the kernel size is 3×3 with stride 1. The second layer is a fully connected layer that consists of 128 neurons. For the previous two layers, we take Leaky Relu [24] as an activation function. The last layer is a fully connected linear layer employed with $N \cdot (N-1)$ neurons, which refers to all possible destination-based forwarding entries that can be selected. There is no activation function for the last layer, and we apply a softmax function to the output for generating the probabilities of all available actions. As for the critic network, the neural network architecture is almost the same as that of the policy network. The only exception is the last layer of the critic network, which only contains one neuron as the estimate of the value function. For hyperparameter settings, we set the initial learning rates α and α_v as 0.0001 with a decay rate of 0.96 every 500 iterations, while the entropy weight β is set to 0.1. Additionally, we set $K = 10\% \cdot N \cdot (N-1)$ to select 10% of total destination-based forwarding entries for routing updates. It turns out that the above hyperparameter settings work well in our evaluations with a stable training process.

V. Traffic Split Ratio Optimization

In this section, we describe how to solve the traffic split ratio optimization problem for the selected entries with different priorities. The notations are listed in Table I.

A. High Priority Traffic Allocation for QoS Provisioning

By default, high priority traffic is evenly distributed among ECMP next hops. Given the selected destination-based forwarding entries from the high priority RL agent, high priority traffic can be split unevenly among available next hops according to the optimized traffic split ratios specified by these entries, as long as there is no forwarding loop in the network.

TABLE I: Notations

$G(V, E)$	network with nodes V and directed links E ($\|V\| = N, \|E\| = M$)
$c_{i,j}$	the capacity of link $\langle i, j\rangle$ ($\langle i, j\rangle \in E$)
$b_{i,j}$	the reserved bandwidth for high priority traffic at link $\langle i, j\rangle$ ($\langle i, j\rangle \in E$)
$l_{i,j}$	the traffic load on link $\langle i, j\rangle$ ($\langle i, j\rangle \in E$)
$t^H_{s,d}$	the high priority traffic demand originated from s destined to d ($s, d \in V$, $s \neq d$)
$t^L_{s,d}$	the low priority traffic demand originated from s destined to d ($s, d \in V$, $s \neq d$)
τ^d_i	the traffic destined to d at node i ($i, d \in V$, $i \neq d$, $\|\{\tau^d_i\}\| = N \cdot (N-1)$)
τ^H_K	a combination of K selected τ^d_i ($\|\tau^H_K\| = K$) for high priority traffic, e.g., τ^9_1 in Fig. 2(b)
τ^L_K	a combination of K selected τ^d_i ($\|\tau^L_K\| = K$) for low priority traffic, e.g., τ^{10}_1 and τ^{10}_3 in Fig. 2(c)
$\tau^H_{N\cdot(N-1)-K}$	the set of remaining τ^d_i for high priority traffic ($\|\tau^H_{N\cdot(N-1)-K}\| = N \cdot (N-1) - K$)
$\tau^L_{N\cdot(N-1)-K}$	the set of remaining τ^d_i for low priority traffic ($\|\tau^L_{N\cdot(N-1)-K}\| = N \cdot (N-1) - K$)
ENH^d_i	the set of ECMP next hops for destination d at node i ($i, d \in V$), e.g., $\text{ENH}^9_1 = \{4\}$ in Fig. 2(a)
$y^d_{i,j}$	the traffic destined to d routed on link $\langle i, j\rangle$ ($d \in V, \langle i, j\rangle \in E$)
$\sigma^d_{i,j}$	the split ratio at node i to node j for the traffic destined to node d ($d \in V, \langle i, j\rangle \in E$), e.g., $\sigma^{10}_{3,5} = 75\%$ in Fig. 2(c)

Given a network $G(V, E)$ with a high priority TM TM^H_t and the selected high priority forwarding entries τ^H_K, our objective is to obtain the optimal weighted split ratios $\{\sigma^d_{i,j}\}$ for the selected $\tau^d_i \in \tau^H_K$, such that the end-to-end delay D is minimized with a loop-free routing. As described in Section II-A, the end-to-end delay can be derived as $D = \sum_{\langle i,j\rangle \in E} l_{i,j}/(c_{i,j} - l_{i,j})$, where a cost function $F_{i,j}(l_{i,j})$ can be used to represent $l_{i,j}/(c_{i,j} - l_{i,j})$. Since the cost function cannot deal with overloaded links [25], we use the piecewise linear approximations of $F_{i,j}(l_{i,j})$ presented in [25], [26] to formulate an LP problem. The high priority traffic rerouting problem is formulated as an optimization problem as follows:

$$\min \sum_{\langle i,j\rangle \in E} F_{i,j}(l_{i,j}) \tag{6a}$$

$$\text{subject to} \quad \sum_{d \in V} y^d_{i,j} = l_{i,j} \qquad i, j : \langle i, j\rangle \in E \tag{6b}$$

$$\sum_{k:\langle k,i\rangle \in E} y^d_{k,i} - \sum_{k:\langle i,k\rangle \in E} y^d_{i,k} = -t^H_{i,d} \quad i, d : \tau^d_i \in \tau^H_K \tag{6c}$$

$$y^d_{i,k} = \begin{cases} \dfrac{\sum_{n:\langle n,i\rangle \in E} y^d_{n,i} + t^H_{i,d}}{|\text{ENH}^d_i|} & \text{if } k \in \text{ENH}^d_i \\ 0 & \text{otherwise} \end{cases} \tag{6d}$$

$$i, d : \tau^d_i \in \tau^H_{N\cdot(N-1)-K},\ k : \langle i, k\rangle \in E$$

$$\sum_{k:\langle k,d\rangle \in E} y^d_{k,d} - \sum_{k:\langle d,k\rangle \in E} y^d_{d,k} = \sum_{s \in V, s \neq d} t^H_{s,d} \quad d \in V \tag{6e}$$

$$y^d_{i,j} \geq 0 \qquad d \in V,\ i, j : \langle i, j\rangle \in E \tag{6f}$$

$$F_{i,j}(l_{i,j}) \geq l_{i,j}/c_{i,j} \qquad i, j : \langle i, j\rangle \in E \tag{6g}$$

$$F_{i,j}(l_{i,j}) \geq 3l_{i,j}/c_{i,j} - 2/3 \qquad i, j : \langle i, j\rangle \in E \tag{6h}$$

$$F_{i,j}(l_{i,j}) \geq 10l_{i,j}/c_{i,j} - 16/3 \qquad i, j : \langle i, j\rangle \in E \tag{6i}$$

$$F_{i,j}(l_{i,j}) \geq 70l_{i,j}/c_{i,j} - 178/3 \qquad i, j : \langle i, j\rangle \in E \tag{6j}$$

$$F_{i,j}(l_{i,j}) \geq 500l_{i,j}/c_{i,j} - 1468/3 \qquad i, j : \langle i, j\rangle \in E \tag{6k}$$

$$F_{i,j}(l_{i,j}) \geq 5000l_{i,j}/c_{i,j} - 16318/3 \qquad i, j : \langle i, j\rangle \in E \tag{6l}$$

(6b) defines the link load as the sum of all high priority traffic routed on the link. (6c)-(6e) are the flow conservation constraints for the selected entries $\tau^d_i \in \tau^H_K$, for the remaining entries $\tau^d_i \in \tau^H_{N\cdot(N-1)-K}$, and at the destinations, respectively. (6f) ensures that the traffic allocation on each link should be non-negative. (6g)-(6l) defines the cost function based on the load and capacity of the link, which is derived from the piecewise linear approximations [25], [26]. We can solve this optimization problem using LP solvers (e.g., Gurobi [27]) to obtain the optimal destination-based high priority traffic allocation $\{y^d_{i,j}\}$, which can be used to derive the optimal weighted traffic split ratios $\{\sigma^d_{i,j}\}$ for the selected high priority entries as follows:

$$\sigma^d_{i,j} = \frac{y^d_{i,j}}{\sum_{k:\langle i,k\rangle \in E} y^d_{i,k}} \quad i, d : \tau^d_i \in \tau^H_K,\ j : \langle i, j\rangle \in E. \tag{7}$$

For the remaining entries not selected by the high priority RL agent (i.e., $\tau^d_i \in \tau^H_{N*(N-1)-K}$), they forward high priority traffic based on default ECMP rules without routing updates.

B. Low Priority Traffic Allocation for Load Balancing

Once high priority traffic is allocated, the corresponding bandwidth is reserved on each link as $\{b_{i,j}\}$ and is no longer available for low priority traffic. The forwarding entries selected by the low priority RL agent can control a portion of low priority traffic in a flexible manner, while most of the low priority traffic is routed by ECMP.

Given a network $G(V, E)$ with a low priority TM TM^L_t and the selected low priority forwarding entries τ^L_K, our objective is to obtain the optimal weighted split ratios $\{\sigma^d_{i,j}\}$ for the selected $\tau^d_i \in \tau^L_K$, such that the MLU U is minimized to achieve good load balancing with a loop-free routing. The low priority traffic rerouting problem is formulated as an optimization problem as follows:

$$\min U \tag{8a}$$

$$\text{subject to} \quad \sum_{d \in V} y^d_{i,j} = l_{i,j} \qquad i, j : \langle i, j\rangle \in E \tag{8b}$$

$$l_{i,j} + b_{i,j} \leq c_{i,j} \cdot U \qquad i, j : \langle i, j\rangle \in E \tag{8c}$$

$$\sum_{k:\langle k,i\rangle \in E} y^d_{k,i} - \sum_{k:\langle i,k\rangle \in E} y^d_{i,k} = -t^L_{i,d} \quad i, d : \tau^d_i \in \tau^L_K \tag{8d}$$

$$y^d_{i,k} = \begin{cases} \dfrac{\sum_{n:\langle n,i\rangle \in E} y^d_{n,i} + t^L_{i,d}}{|\text{ENH}^d_i|} & \text{if } k \in \text{ENH}^d_i \\ 0 & \text{otherwise} \end{cases} \tag{8e}$$

$$i, d : \tau^d_i \in \tau^L_{N\cdot(N-1)-K},\ k : \langle i, k\rangle \in E$$

$$\sum_{k:\langle k,d\rangle \in E} y^d_{k,d} - \sum_{k:\langle d,k\rangle \in E} y^d_{d,k} = \sum_{s \in V, s \neq d} t^L_{s,d} \quad d \in V \tag{8f}$$

$$y^d_{i,j} \geq 0 \qquad d \in V, i, j : \langle i, j \rangle \in E \tag{8g}$$

Most of the constraints in (8) are similar to those in (6). The major difference is that the delay-centric constraints (6g)-(6l) are substituted with an MLU-centric constraint (8c), which defines MLU as the utilization of the most congested link in the network with consideration of reserved bandwidth. We can use an LP solver to find the optimal destination-based low priority traffic allocation $\{y^d_{i,j}\}$, and then convert it to the optimal traffic split ratios $\{\sigma^d_{i,j}\}$ for the selected low priority entries $\tau^d_i \in \tau^L_K$ based on Eq. (7). For safety concerns, we leverage the techniques in [28] to detect and eliminate possible forwarding loops in the optimized routing strategy before deploying the updated entries at the underlying switches.

VI. Performance Evaluation

In this section, we conduct extensive simulations on real-world network topologies to evaluate the performance of QoS-RL in terms of end-to-end delay and MLU.

A. Simulation Setup

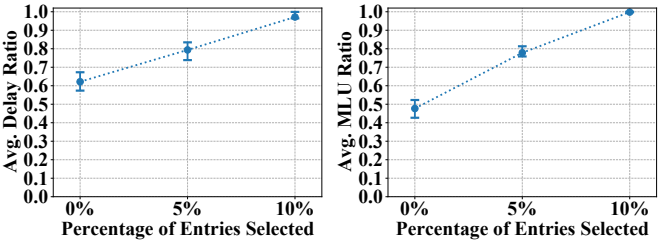
WAN Topologies. We evaluate QoS-RL in four different real-world network topologies, including three ISP networks (EBONE, Sprintlink, Tiscali) from Rocketfuel [29] and the Germany50 network [30]. The number of nodes and directed links of each topology is listed in Table II. For Rocketfuel topologies, we infer link capacities as the inverse of provided link weights, which is commonly adopted in literature [28]. For Germany50, we set all link weights to 10 and configure link capacities based on the degrees of their connected nodes (if both degrees < 4, set to 5 Gbps; otherwise, 10 Gbps) [31].

Traffic Generation. To generate TMs that match real-world traffic patterns, we refer to the traffic statistics measured in production WANs. As reported in Baidu's private WAN, high priority traffic usually exposes diurnal patterns with strong temporal correlations [16]. Therefore, we leverage the Modulated Gravity Model (MGM) [32], [33] to generate a time series of high priority TMs for the Rocketfuel topologies, as MGM can efficiently reflect the cyclical nature of high priority traffic with temporal correlations. By adjusting the spatial variances and peak-to-mean ratios in MGM, we generate 50 TMs with large traffic variations and 50 TMs with small traffic variations for each ISP network as the high priority TM training dataset. We also generate the test dataset in a similar way with different MGM parameters. For low priority traffic, the traffic pattern is relatively stable [6]. Thus, we generate a series of low priority TMs using a uniform model [33]. For each ISP network, there are 100 uniform TMs for training QoS-RL, and we generate another 100 uniform test TMs with different model parameters for evaluation. As for the Germany50 network, limited real TMs with temporal patterns in one day are available [34]. Thus, we take these measured TMs as high priority TMs and split them into the training set and test set in an 85%/15% manner. We also generate the same number of low priority uniform TMs for training and testing.

Baselines. In addition to QoS-RL, we evaluate three different destination-based TE solutions for comparison. The first one is ECMP [17], which serves as the default static routing to forward traffic evenly among ECMP next hops. Another scheme is Weighted ECMP (W-ECMP), which extends ECMP by allowing weighted traffic splitting among ECMP next hops with split ratios optimized by LP [4]. Besides, we also evaluate a rule-based heuristic called Top-K to update the top K destination-based forwarding entries that forward the most traffic under ECMP routing. Similar to QoS-RL, Top-K optimizes the split ratios of these K entries for high priority traffic and low priority traffic with LP formulations (6) and (8), respectively. The only difference between QoS-RL and Top-K is the selection criteria of the K entries to be updated.

TABLE II: WAN Topologies for Evaluation

Topology	# Nodes	# Directed Links
EBONE (Europe)	23	76
Sprintlink (US)	44	166
Tiscali (Europe)	49	172
Germany50	50	176



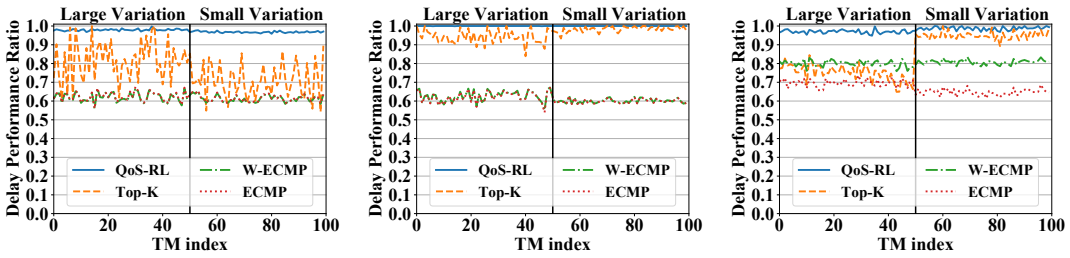
(a) End-to-end delay vs. K. (b) MLU performance vs. K.

Fig. 3: Evolution of average delay ratio and MLU ratio with an increasing number of entry updates K in the EBONE network. The error bar represents the highest/lowest performance ratio.

Metrics. For QoS-RL and the baseline methods, we evaluate the delay performance ratio PR_D in Eq. (1) to demonstrate the QoS provided to high priority traffic. Meanwhile, the MLU performance ratio PR_U in Eq. (2) is computed for each TE solution to reveal the overall load balancing performance in the network after allocating low priority traffic. The higher the ratios, the better the QoS and load balancing performance.

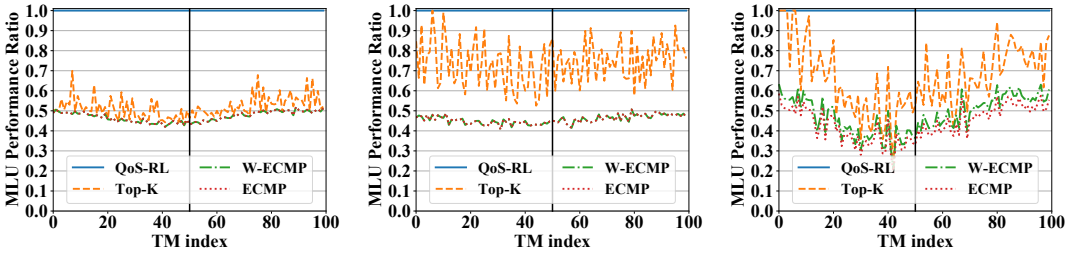
B. Number of Entry Updates

In QoS-RL's design, one critical aspect is to determine a proper number of entry updates K to achieve efficient QoS provisioning and good load balancing with low management overhead. To find a proper setting for K, we train several QoS-RL models with different K settings and evaluate their performance. Fig. 3 depicts the average delay performance ratio and MLU performance ratio achieved by QoS-RL on the test set of the EBONE network with an increasing K, where 0% of entry updates is equivalent to ECMP. Compared to ECMP, 5% of entry updates can provide a 17.2% average delay ratio improvement and a 30.1% average MLU ratio improvement, which demonstrates the feasibility of improving network performance by only updating a few entries. When updating 10% of total entries, QoS-RL can achieve an average delay ratio of 97.3% and an average MLU ratio of 100%, which is sufficient for good QoS provisioning and load balancing. Since the results in the other three networks are similar, we set K to 10% of total entries in the following experiments.



(a) EBONE network. (b) Sprintlink network. (c) Tiscali network.

Fig. 4: End-to-end delay performance comparison in the three ISP networks with high priority TMs.



(a) EBONE network. (b) Sprintlink network. (c) Tiscali network.

Fig. 5: Load balancing performance comparison in the three ISP networks with low priority TMs.

C. QoS Provisioning for High Priority Traffic

Fig. 4 shows the delay performance ratio that each scheme achieves on the high priority test TMs of the three ISP networks, where large traffic variations usually lead to more frequent performance fluctuations compared to small traffic variations. As a default routing strategy, ECMP performs the worst in the three networks. Even though ECMP forwards traffic along the shortest paths with potentially lower latency, it may over-utilize the link resources and result in severe network congestion with degraded QoS. In addition, there is no routing update for ECMP to accommodate traffic dynamics. Therefore, the overall delay performance of ECMP is far from optimal. Compared to ECMP, W-ECMP can support flexible split ratios among ECMP next hops and bypass the restriction of equal splitting. As shown in Fig. 4(c), W-ECMP can improve the average delay performance ratio to 80.4% in the Tiscali network. However, W-ECMP is still constrained by shortest paths and thus cannot achieve better delay performance than ECMP in the EBONE and Sprintlink networks.

By selecting the top 10% destination-based forwarding entries that control the most traffic under ECMP routing, Top-K can optimize the traffic split ratios for these entries with LP to reduce network delay. As depicted in Fig. 4(b), Top-K is able to achieve above 90% of optimal delay performance in most high priority traffic scenarios with a few exceptions in the Sprintlink network. However, the delay performance of Top-K is unstable in other networks. As shown in Fig. 4(a), we can observe frequent delay performance fluctuations of Top-K in the EBONE network. In the Tiscali network, Top-K is also outperformed by W-ECMP under large traffic variations. It turns out that simple rule-based heuristic methods cannot effectively adapt to dynamic network conditions. In contrast, QoS-RL achieves promising end-to-end delay performance in all three ISP networks by updating the same number of forwarding entries as Top-K, which demonstrates the intelligence of the high priority RL agent to adaptively select a good combination of 10% entries under different traffic variations. In Fig. 4, QoS-RL provides at least 94.8% of optimal delay performance for all high priority test TMs with an average delay performance ratio of 97.3%-99.9% in the three networks. As a result, QoS-RL is capable of providing good QoS to high priority traffic with close-to-optimal delay performance.

D. Load Balancing for Low Priority Traffic

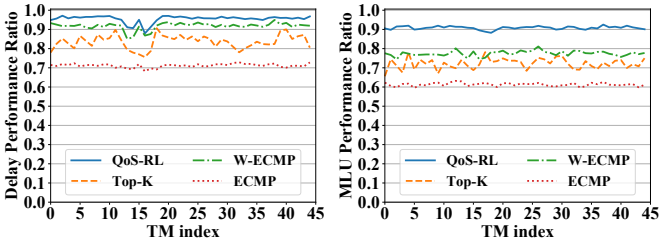
Fig. 5 shows the MLU performance ratio of each TE scheme in the three ISP networks after allocating low priority traffic. Similarly, we can observe the sub-optimal load balancing performance of ECMP and W-ECMP as well as the fluctuating MLU performance ratio of Top-K. Surprisingly, QoS-RL achieves an MLU performance ratio of $PR_U = 1$ for all low priority test TMs in the three networks, which greatly outperforms other baseline methods. As demonstrated in Fig. 5(a), QoS-RL can achieve 48.2% average load balancing performance improvement compared to the best-performing Top-K method in EBONE. Instead of updating all forwarding entries with severe service disruption, QoS-RL can provide optimal load balancing performance by intelligently updating 10% of forwarding entries identified by the low priority RL agent. Overall, QoS-RL is able to satisfy the QoS requirements of high priority traffic and maintain good load balancing performance by efficiently allocating low priority traffic.

E. Generalization to Unseen Traffic Scenarios

To further demonstrate the generalization capability of QoS-RL, we take the measured traffic traces from the Germany50 network as high priority TMs and evaluate QoS-RL in real-world unseen traffic scenarios. Fig. 6 presents the end-to-end delay and MLU performance of different TE schemes on each test TM in the Germany50 network. Compared to the generated TMs in the three ISP networks, the real Germany50

TABLE III: Offline Training Time of Different RL Agents

Topology	High Priority	Low Priority
EBONE	6.5 hours	4 hours
Sprintlink	8 hours	6 hours
Tiscali	9 hours	7 hours
Germany50	10.5 hours	8.5 hours



(a) End-to-end delay. (b) MLU performance.

Fig. 6: Delay and MLU performance comparison in Germany50.

TMs appear to be more dynamic, which could be challenging for QoS-RL to extract real-world traffic patterns. However, QoS-RL can still generalize well to these unseen traffic scenarios. As shown in Fig. 6(a), QoS-RL achieves 95.5% of optimal delay on average in the unseen high priority test TMs. Although there is an exception (the 16th test TM) where QoS-RL only provides 87.9% of optimal delay, it still outperforms other baseline methods in all unseen traffic scenarios.

Since the real Germany50 TMs are highly dynamic, the resulting bandwidth allocations and reservations for these high priority TMs could vary a lot. In this situation, the low priority RL agent is making decisions under highly dynamic network conditions even though the uniform low priority TMs are relatively stable. Unlike the previous three ISP networks, QoS-RL cannot provide the same load balancing performance as the optimal solution in the Germany50 network. However, as shown in Fig. 6(b), QoS-RL is still capable of maintaining near-optimal load balancing performance in a more dynamic environment with an average MLU performance ratio of 90.8%, which outperforms the best-performing baseline by 13.3% and saves 90% of entries compared to the optimal solution. In conclusion, QoS-RL exhibits strong generalization capability in unseen real-world traffic scenarios with efficient QoS provisioning and good load balancing performance.

F. Overhead Analysis

Table III shows the training time of different RL agents in the four networks, where the training procedure is conducted in a high-performance computing cluster with 21 CPU cores and 32 GB memory for parallel training (i.e., each CPU core serves as an actor to experience a subset of training TMs). It usually takes a longer time to train QoS-RL in larger networks with a larger input state and action space. Also, a high priority RL agent requires a longer training time compared to a low priority RL agent. This is because the computation overhead of solving optimal high priority traffic allocation (6) is longer than that of the low priority LP (8). Since the reward signals are relying on the performance metrics derived from LP, it is reasonable for the high priority RL agent to experience a longer training process than the low priority RL agent. Overall, it takes 10.5-19 hours in total to train QoS-RL in the four networks. Note that the overall training process is conducted in an offline manner, and there is no need to frequently retrain a QoS-RL model since it exhibits good generalization capability.

TABLE IV: Average Online Execution Time in the Four Networks

Topology	High Priority (sec)		Low Priority (sec)	
	QoS-RL*	Optimal	QoS-RL*	Optimal
EBONE	0.011+0.86	5.06	0.004+0.042	0.260
Sprintlink	0.039+11.31	55.13	0.038+0.160	2.03
Tiscali	0.044+16.84	73.17	0.045+0.214	2.63
Germany50	0.243+23.90	85.40	0.244+0.254	3.36

*RL inference time + LP solving time.

Table IV shows the computation overhead of QoS-RL during online deployment, which is evaluated in our simulation environment in a Linux server (4-core 3.4 GHz CPU and 16 GB memory) with Gurobi optimizer [27]. As the network size grows, the dimensions of the input TM and the output layer of the policy network become larger, which leads to a longer inference time. But, the overall inference process is still time-efficient since the RL agent can identify 10% of important entries in less than 1 second on average. For traffic split ratio optimization, there are more constraints in the high priority LP (6) compared to the low priority LP (8). Thus, it takes 23.9 seconds on average to solve the high priority LP in the Germany50 network, while the low priority LP is relatively time-efficient. Such execution time should be acceptable since the Germany50 TMs are measured at 5-minute intervals. Moreover, the computation complexity has been greatly reduced by QoS-RL with only 10% of entry updates. As shown in Table IV, QoS-RL can achieve considerable time savings compared to optimal LP that updates all entries in the network, which enables QoS-RL to quickly react to traffic changes and results in lower management overhead (e.g., communication, deployment, storage) with mitigated service disruption.

VII. Related Work

Traditional TE usually relies on routing protocols to forward the network traffic. [3], [25], and [35] leverage Open Shortest Path First (OSPF) and ECMP to balance link utilization by dynamically adjusting link weights for shortest paths. Weighted ECMP [4] extends ECMP by allowing weighted traffic splitting among ECMP nodes to improve network performance. These works are usually constrained by OSPF convergence speed and shortest path routing. Another line of work focuses on Multiprotocol Label Switching (MPLS) networks for TE purposes. By solving an optimization problem, [1] and [2] can obtain the explicit paths of flows and distribute the flows accordingly. However, they cannot effectively adapt to dynamic traffic conditions without timely routing updates.

The emergence of SDN [7] provides a flexible way for TE to perform fine-grained traffic control. Dynamic hybrid routing [9] dynamically re-balances traffic with SDN to achieve good load balancing. SMORE [36] generates a set of preconfigured paths and adaptively adjusts the sending rates to accommodate traffic changes. Under the context of SDN-based TE, QoS provisioning for different applications is also considered an objective of cloud service providers. While aiming at maximizing network throughput and achieving high utilization in

inter-datacenter WANs, Google [5] and Microsoft [6] assign different priorities to traffic flows based on their QoS requirements and allocate bandwidth in priority order to ensure good service quality. However, as we discussed in Section I, these approaches could lead to a negative impact on network operations with considerable management overhead.

Recently, Machine Learning (ML) techniques have been adopted for TE design. Valadarsky et al. [37] use RL to obtain a good set of link weights for generating routing strategies. Xu et al. [38] exploit RL to optimize different performance metrics, including throughput and delay. Besides, there are some existing ML-based TE methods for QoS provisioning. QoS-aware Adaptive Routing [39] leverages RL to design a distributed three-level control plane for minimizing signaling delay. DRL-OR [40] employs deep RL for per-flow routing decisions to provide good QoS for different applications in an online routing scenario. However, these methods adopt fine-grained per-flow traffic control mechanisms for QoS provisioning and neglect the negative impact on management overhead.

VIII. Conclusion

To achieve good QoS provisioning and load balancing with consideration of management overhead and service disruption, we propose QoS-RL, an RL-based TE solution that intelligently identifies a few destination-based forwarding entries for rerouting different priorities of traffic flows. QoS-RL adopts a two-step optimization approach to allocate high priority traffic at first with good QoS provisioning, and then effectively reroutes a small portion of low priority traffic to balance the link loads with reduced overhead. Extensive simulation results show that QoS-RL achieves near-optimal end-to-end delay performance and load balancing performance in unseen traffic scenarios by only updating 10% of total entries.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62002019, Zhejiang Lab Open Research Project under Grant K2022QA0AB02, Song-Shan Laboratory Fund under Grant YYJC022022009, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] Y. Wang and Z. Wang, "Explicit routing algorithms for internet traffic engineering," in *IEEE ICCCN*, 1999.

[2] E. D. Osborne and A. Simha, *Traffic engineering with MPLS*. Cisco Press, 2002.

[3] J. Chu and C.-T. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns," *IEEE/ACM ToN*, 2009.

[4] J. Zhang, K. Xi, L. Zhang, and H. J. Chao, "Optimizing network performance using weighted multipath routing," in *IEEE ICCCN*, 2012.

[5] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM CCR*, 2013.

[6] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM*, 2013.

[7] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, 2008.

[8] M. Moradi, Y. Zhang, Z. Morley Mao, and R. Manghirmalani, "Dragon: Scalable, flexible, and efficient traffic engineering in software defined isp networks," *IEEE JSAC*, 2018.

[9] J. Zhang, K. Xi, M. Luo, and H. J. Chao, "Dynamic hybrid routing: Achieve load balancing for changing traffic demands," in *IEEE IWQoS*, 2014.

[10] S. Q. Zhang *et al.*, "Tcam space-efficient routing in a software defined network," *Computer Networks*, 2017.

[11] A. X. Liu, C. R. Meiners, and E. Torng, "Tcam razor: A systematic approach towards minimizing packet classifiers in tcams," *IEEE/ACM ToN*, 2009.

[12] K. Kannan and S. Banerjee, "Compact tcam: Flow entry compaction in tcam for power aware sdn," in *Distributed Computing and Networking*. Springer, 2013.

[13] A. R. Curtis *et al.*, "Devoflow: scaling flow management for high-performance networks," *ACM SIGCOMM CCR*, 2011.

[14] S. Kotachi, T. Sato, R. Shinkuma, and E. Oki, "Multicast routing model to minimize number of flow entries in software-defined network," in *APNOMS*, 2019.

[15] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *ACM SIGCOMM*, 2018.

[16] Z. Wang *et al.*, "Examination of wan traffic characteristics in a large-scale data center network," in *ACM IMC*, 2021.

[17] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," *IETF RFC 2991*, November 2000.

[18] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[19] W. Reda *et al.*, "Path persistence in the cloud: A study of the effects of inter-region traffic engineering in a large cloud provider's network," *ACM SIGCOMM CCR*, 2020.

[20] R. Carpa, M. D. de AssunçÃo, O. Glück, L. LefÈvre, and J.-C. Mignot, "Evaluating the impact of sdn-induced frequent route changes on tcp flows," in *IEEE CNSM*, 2017.

[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.

[22] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.

[23] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *USENIX OSDI*, 2016.

[24] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[25] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE JSAC*, 2002.

[26] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in ospf routing," *Journal of combinatorial optimization*, 2002.

[27] Gurobi Optimization. [Online]. Available: https://www.gurobi.com

[28] J. Zhang, K. Xi, and H. J. Chao, "Load balancing in ip networks using generalized destination-based multipath routing," *IEEE/ACM ToN*, 2015.

[29] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM CCR*, 2002.

[30] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *INOC, Spa, Belgium*, 2007.

[31] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Federated traffic engineering with supervised learning in multi-region networks," in *IEEE ICNP*, 2021.

[32] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," *ACM SIGCOMM CCR*, 2015.

[33] TMgen: Traffic Matrix Generation Tool. [Online]. Available: https://tmgen.readthedocs.io/en/latest/

[34] SNDlib. [Online]. Available: http://sndlib.zib.de/home.action

[35] K. Holmberg and D. Yuan, "Optimization of internet protocol network design and routing," *Networks: An International Journal*, 2004.

[36] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *USENIX NSDI*, 2018.

[37] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *ACM HotNets*, 2017.

[38] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM*, 2018.

[39] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE SCC*, 2016.

[40] C. Liu, M. Xu, Y. Yang, and N. Geng, "Drl-or: Deep reinforcement learning-based online routing for multi-type service requirements," in *IEEE INFOCOM*, 2021.