MARINA: Multi-Agent Reinforcement Learning-Based Routing with Intelligent Network Adaptation in SDN

Hamed Nazari
Department of Computer Science
University of Regina
Regina, Canada
hamednazari@uregina.ca

Nashid Shahriar
Department of Computer Science
University of Regina
Regina, Canada
nashid.shahriar@uregina.ca

Piotr Boryło
Institute of Telecommunications
AGH University of Krakow
Krakow, Poland
piotr.borylo@agh.edu.pl

***Abstract*—Optimizing routing in Software-Defined Networks (SDNs) to meet Quality of Service (QoS) demands presents notable challenges, particularly with traditional methods that struggle with the dynamic nature of SDNs. Existing techniques often fail to adapt efficiently to varying traffic patterns, resulting in suboptimal network performance. Furthermore, the existing routing methods typically prioritize either the infrastructure provider or service provider benefits, neglecting a balanced approach that ensures efficient utilization of network resources while meeting QoS metrics for clients, such as guaranteed latency and throughput. This paper proposes a multi-agent Deep Reinforcement Learning (DRL)-based QoS-aware routing solution in SDN environments with intelligent network adaptation, called MARINA. MARINA dynamically adjusts routing paths for both existing and new incoming traffic flows in the network. Moreover, MARINA simultaneously optimizes network resource usage and meets QoS constraints such as latency and guaranteed throughput in an SDN. MARINA is extensively evaluated on real-world GEANT2 network topology. The results indicate that our algorithm outperforms Open Shortest Path First (OSPF), Equal Cost Multiple Path (ECMP), and single-agent DRL in key performance metrics. Specifically, MARINA meets 51.4%, 29.4%, and 11.1% more QoS requirements than OSPF, ECMP, and single-agent DRL-based approaches, respectively.**

***Index Terms*—Software-defined Networking, Routing, Deep Reinforcement Learning, Network Adaptivity.**

I. Introduction

Software-Defined Networking has emerged as a transformative approach to managing and optimizing network resources. By decoupling the control plane from the data plane, SDN offers a centralized management system that enhances flexibility, scalability, and the ability to dynamically respond to varying network conditions [7]. However, ensuring Quality of Service in SDN remains a significant challenge due to the dynamic nature of modern network environments and fluctuating traffic patterns [10]. Traditional routing methods often fall short in addressing these challenges as they typically lack the adaptability required to maintain QoS guarantees, especially under conditions of high variability in network demands [7], [14].

Recent advances have highlighted the potential of machine learning-based methods to enhance SDN routing efficiency. DRL is an ideal approach to address decision-making and optimization problems such as routing [11]. In recent years, a growing trend of DRL approaches for routing in SDN has been observed [14]. However, existing DRL-based strategies are not generalizable and use limited information during routing. One of the key challenges is the poor support for QoS and the lack of adaptability to traffic variability in SDN, which restricts their effectiveness in dynamic network environments. Among these, DRL algorithms such as Deep Q-Networks (DQN) have shown promise due to their ability to learn optimal routing strategies in complex environments [11]. Nevertheless, current DQN-based approaches often focus primarily on either the infrastructure provider side or the service provider side, neglecting an efficient way of ensuring effective utilization of network resources while meeting QoS metrics for clients such as guaranteed latency and throughput, which can significantly impact overall network performance [3].

To address these challenges, this paper proposes a multi-agent DRL-based dynamic routing solution in SDN with QoS optimization that efficiently reroutes existing active traffic flows and properly routes incoming traffic while satisfying throughput and delay constraints and optimizing overall network resource usage. We develop a Multi-Agent reinforcement learning-based Routing with Intelligent Network Adaptation (MARINA) as a QoS-aware SDN routing algorithm that leverages a cooperative multi-agent DRL. The proposed MARINA solution is designed to dynamically allocate the best paths for incoming traffic flows while making QoS requirements more responsive and capable of adapting to diverse network conditions. Additionally, our proposed solution can enhance the network's ability to accommodate future requests by efficiently utilizing network resources.

The main contributions of this paper are as follows: 1) We introduce a multi-agent DRL-based QoS-aware SDN routing algorithm that applies a cooperative multi-agent DRL solution to enhance routing decisions; 2) MARINA considers both service provider requirements such as multiple QoS constraints, including guaranteed throughput and delay, and infrastructure provider requirements in terms of optimizing the

network resource usage; 3) Our proposed solution not only identifies the optimal paths for incoming traffic flows but also dynamically re-routes previously established traffic flows in real time, ensuring efficient network resource utilization and adaptibility.

The rest of the paper is organized as follows: In Section II, we discuss related work. We allocated Section III to the system model and problem formulation. Section VI and Section V are dedicated to describing the methodology and evaluating the performance of MARINA, respectively. Finally, Section IV concludes and outlines future work.

II. Related Works

Routing optimization in traffic engineering is a well-established topic in the literature [11]. The purpose of routing optimization algorithms is to control the behavior of the route for data transmission to improve the network performance and meet QoS requirements. There are a wide variety of studies in the network routing realm, which are generally based on analytical optimization [16], local-search heuristics [8], or machine learning techniques [2], [3], [7]. In recent years, the development of routing optimization algorithms for SDN with machine learning-based techniques has received attention [11].

Within this context, DRL has emerged as a promising approach for optimizing routing decisions in SDN environments [3], [7]. Although single-agent RL solutions have been extensively applied in the SDN routing realm, these methods often struggle with scalability and generalizability to adapt to dynamic or large-scale networks, as the action space expands exponentially with the network size. Recent solutions such as DRSIR [3] address some of these challenges by leveraging DQN; however, single-agent solution has limited adaptibility in highly dynamic network environments and handling various types of traffic flows.

Other approaches focus on energy efficiency by dynamically adjusting routing paths to reduce energy consumption without compromising QoS [13]. Most current DRL-based approaches primarily focus on either service provider benefits or infrastructure provider benefits, often neglecting the other part. This narrow focus can lead to suboptimal network performance in scenarios where multiple QoS requirements must be balanced or focus is on efficiently utilizing network resources and ignoring QoS metrics. Additionally, as the size and complexity of the network increase, the state and action spaces in DRL expand exponentially, making it difficult to learn efficiently and converge to an optimal policy [14]. Despite these advancements, existing DRL-based routing methods in SDN face several challenges and gaps [7], [10].

Rischke et al. [12] proposed a flow routing solution which is quite different from traditional routing solutions. The authors presented a model-free Q-learning-based RL solution in SDN named QR-SDN. QR-SDN operates on a per-flow basis which means that it preserves the flow integrity while providing multipath routing. Hence, QR-SDN helps reduce out-of-order packets and complexities to some extent. Casas-Velasco et al. [3] used DRL-based approaches to improve routing adaptation to traffic variability and effectively support QoS required in SDN. In [3], DRL and SDN Intelligent Routing (DRSIR) is presented as a routing optimization approach to overcome the limitations of RL-based routing. DRSIR provides intelligent, efficient and proactive routing paths by considering path-state metrics. Considering path-state metrics in routing has enabled DRSIR to adapt to dynamic traffic changes.

Dhandapani et al. [4] proposed CoopAI-Route, a Multi-Agent RL (MARL) framework designed for QoS-aware routing in multi-domain SDNs. By leveraging hierarchical SDN architecture, CoopAI-Route integrates message-passing neural networks (MPNNs) with the twin-delayed deep deterministic policy gradient (TD3) algorithm to enable intelligent and cooperative routing decisions. In their work, the main focus is on multi-domain structures, and one agent is responsible for a certain domain of the network.

In summary, existing DRL-based routing approaches, such as DRSIR and CoopAI-Route, focus on improving adaptability, scalability, or specific QoS metrics like delay or energy efficiency but often fail to holistically address the balance between current and previous traffic flows or dynamic network conditions. Unlike these methods, we propose MARINA that introduces a novel cooperative multi-agent framework that simultaneously finds the best routing path for incoming traffic flow and reroutes currently active traffic flows in the network. This enables MARINA to adapt to changing network conditions, optimize resource utilization and comprehensively meet the QoS requirements of diverse traffic flows.

III. System Model and Problem Definition

The proposed framework for optimizing SDN routing leveraging multi-agent reinforcement learning is depicted in Fig.1. This figure provides an overview of the system architecture and highlights the interactions between various modules, which work in tandem to achieve efficient network routing under dynamic traffic conditions and QoS constraints. The core of this architecture is BNNetSimulator [6]. BNNetSimulator is not inherently designed to operate within SDN environments; thus, we developed custom modules and a DRL environment integrated into the SDN controller, which will be further detailed in Section V. Mininet [1] was another option considered for the SDN environment. We found Mininet with a critical limitation in simulating transmission and queuing latencies and packet drops over some experiments. Hence, we utilize BNNetSimulator which is a packet-level simulator and aligns with latency and packet drop network behaviour.

BNNetSimulator requires three configuration text files as inputs to run each round of simulation, including a routing matrix file that contains routing paths between each pair of source and destination nodes; a topology graph file, which contains the graph of network topology; and a traffic matrix file, which contains all existing traffic flows' characteristics between sources and destinations in each point in time. To

[1] https://mininet.org/

clarify, the traffic matrix file specifies all the required characteristics of flows in the network. For instance, one row of the traffic matrix file, namely a traffic flow string, specifies a traffic flow with its source, destination, bandwidth demand, and packet size.

We configure and feed the network topology, routing matrix, and traffic matrix files to BNNetSimulator in each round of simulation. It then generates detailed statistics for each execution round. Each round, considered as a timestep, involves adding new traffic demands or removing expired traffic demands from the traffic matrix file and updating the routing configuration to reflect changes in network conditions. The simulator then processes these updates and outputs comprehensive statistics, capturing the impact of the current routing and traffic on network performance. This iterative process allows for continuous evaluation and optimization of routing decisions based on real-time feedback from the network.

In this paper, we define the network topology as $G = (V, E, C)$, representing the SDN controller's viewpoint of the network using real-time statistics to address the problem of jointly efficient network resource utilization and QoS-aware routing optimization. Here, the set of nodes is considered as $V = \{v_1, v_2, \ldots, v_i, \ldots, v_{|V|}\}$, and the set of links (edges) is considered as $E = \{e_{(i,j)} | \forall v_i, v_j \in V\}$ in graph G. According to this definition, a node i is denoted by v_i, and $e_{(i,j)}$ is the link between v_i and v_j. Meanwhile, let $|x|$ denote the number of elements in a vector x. $C = \{c_{(i,j)} | \forall v_i, v_j \in V\}$ is the unoccupied bandwidth capacity of the link $e_{(i,j)}$. At each timestep t, the set of characteristics of active traffic flow requests is shown as $T = \{\tau_1, \tau_2, \ldots, \tau_i\}$ where $i \leq t$. This indicates that the number of active traffic flows at any timestep depends on the expiration of their respective lifetimes. Here, $\tau_i = (s, d, bw, lat, los, tt, pkt, lt)$ represents the i-th traffic flow request, s and d are the source and destination nodes of the request, respectively. Here, bw, lat, los, and tt indicate the requested bandwidth, the tolerable latency, the tolerable packet loss ratio, and the type of traffic, respectively, for the request. All QoS requirments including the requested bandwidth, tolarable latency and packet loss ratio vary for different types of traffic. Also, pkt indicates the length of the packet and lt indicates the lifetime of traffic.

IV. Methodology

Although we implemented our solution on top of BNNetSimulator, the simulator is not designed to directly interact with DRL agents. To make this happen, we have developed a framework containing a routing, a network environment, and a traffic generator module, responsible for both feeding the input files (topology, routing matrix, and traffic matrix) and preparing the environment wrapper module for communicating with DRL agents. The simulator also generates network/link level and End-to-End(E2E) level statistics. We parse and record these statistics in the Network/E2E metrics repository and send them to the environment wrapper module.

We have also developed a customized graph-based environment wrapper tailored for routing optimization. It encapsulates the network environment module and facilitates seamless communication with network environment, routing, and traffic generator modules. The Environment Wrapper dynamically constructs the state space, action space, and reward function essential for training and deploying the DRL agent. The DRL agent and the abstracted environment within the wrapper are both implemented on the SDN controller's control plane, ensuring efficient coordination and decision-making.

A. State Space

Considering the traffic generator and all the network and E2E metrics stored in the repository, the state s_t in our environment is a combination of incoming traffic characteristics such as s, d, tt and the current state of candidate paths in the network. The first part of each state s_t in a topology with n nodes is the traffic characteristics, and the second part of the state is a vector of all candidate paths utilization, E2E latency, and packet loss ratio.

B. Action Space

The DRL agent selects the best path index among the finite set of predefined candidate paths. Therefore, the size of the action space is equal to $|A| = i$, where $A = P = \{p_1, p_2, \ldots, p_i\}$ indicates the list of possible actions, which are predefined candidate i-shortest paths. Accordingly, i determines the number of possible routing decisions available to the agent.

C. Reward Function

Reward function is the key part of any RL problem, as it guides the learning process of the agent with proper positive or negative feedback on the agent's actions. In this paper, we define the reward function as a function of current traffic requirements, such as tolerable latency d_{tt} and tolerable packet loss ratio l_{tt}, and the resulted statistics from the selected path p_i, including the latency $delay_{p_i}$, packet loss ratio $loss_{p_i}$, and utilization of the selected path u_{p_i}. u_{p_i} denotes the utilization level of the most congested queue at the source node of a link in the path p_i and is quantified in Eq. 1, where u_{link} is the utilization of an individual link.

$$u_{p_i} = \max_{link \in p_i} (u_{link}) \tag{1}$$

Therefore, the reward function attempts to minimize the maximum link utilization by selecting the path with the lowest maximum utilized link and helps balance traffic across the network. In addition, the reward function encourages the agent to select paths that meet the latency and loss tolerance requirements. Eq. 2 represents the total reward signal at each time step that consists of components of utilization, latency, and loss.

$$R_t = r_{util} + r_{lat} + r_{loss} \tag{2}$$

where, r_{util} defined in Eq. 3:

$$r_{util} = \begin{cases} +1 & u_{p_i} < 0.5 \\ 1 - u_{p_i} & 0.5 \leq u_{p_i} < 0.95 \\ -1 & u_{p_i} \geq 0.95 \end{cases} \tag{3}$$

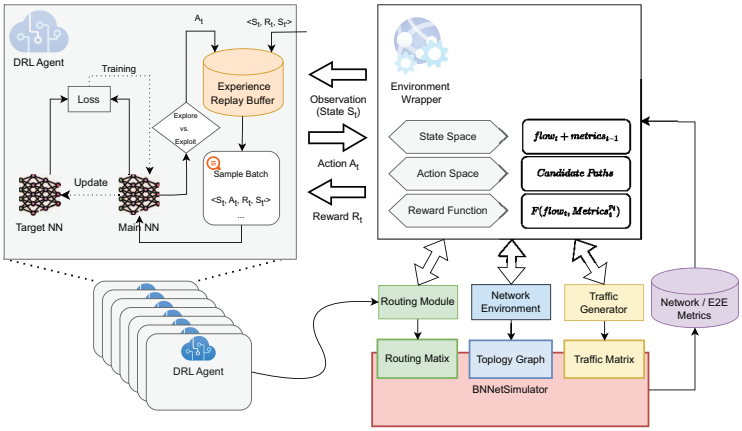


Fig. 1: Overview of the proposed framework

And r_{lat} defined in Eq. 4:

$$r_{lat} = \begin{cases} max(0, \frac{d_{tt} - delay_{p_i}}{d_{tt}}) & delay_{p_i} < d_{tt} \\ -\frac{delay_{p_i} - d_{tt}}{d_{tt}} & d_{tt} \leq delay_{p_i} < 2d_{tt} \\ -1 & Otherwise \end{cases} \quad (4)$$

And r_{loss} defined in Eq. 5:

$$r_{loss} = \begin{cases} max(0, \frac{l_{tt} - loss_{p_i}}{l_{tt}}) & loss_{p_i} < l_{tt} \\ -\frac{loss_{p_i} - l_{tt}}{l_{tt}} & l_{tt} \leq loss_{p_i} < 2l_{tt} \\ -1 & Otherwise \end{cases} \quad (5)$$

r_{util} rewards the RL agent for selecting paths with lower maximum utilized links, which leads to balancing traffic across the network. Similarly, r_{lat} and r_{loss} encourage the agent to select paths that adhere to latency and packet loss tolerance requirements according to different traffic types.

This reward function formulation directs the agent towards selecting the best network paths while optimizing network resources usage through incentivising paths with lower maximum utilized links and meeting QoS requirements such as latency and packet loss ratio. Ultimately, the total reward R_t is the sum of these three components r_{util}, r_{lat}, and r_{loss}, each of which is bounded between $[-1, 1]$. This reward structure allows the agent to adapt dynamically to varying network conditions and traffic demands. By combining positive reinforcement for compliance with QoS metrics and penalties for deviations, the reward function ensures a balance between meeting individual traffic requirements and optimizing overall network performance.

D. Single Agent DQN Routing

According to the environment and due to the very large state space, analyzing the quality of every state-action pair, as commonly used in tabular RL approaches, becomes impractical. To cope with this challenge, employing Deep Learning (DL) has demonstrated significant potential in estimating the Quality-value (Q-value) of state-action pairs. In this paper, we developed DQN-based agents with two main and target Neural Networks (NNs) to enhance the training process. Moreover, we applied prioritized experience replay [15] to accelerate the training process. DQN is particularly well-suited for environments with large state spaces and discrete action spaces [18].

As depicted in Fig. 1 and considering a single DQN agent, the training process of the DQN agent begins with receiving the current environment state S_t, including the incoming traffic flow and all the candidate paths' metrics from the environment wrapper module. Based on this state, the DQN agent needs to select an action A_t using decaying ϵ-greedy exploration-exploitation action selection policy. The objective of this policy in the DQN is to achieve a balance between exploration and exploitation by selecting either a random action (exploration) or the action recommended by the Q-network (exploitation). Initially, the value of $\epsilon \in [0, 1]$ is fixed to a high level in order to promote exploration. The adjustable parameter ϵ in decaying ϵ-greedy policy specifies the level of exploitation of the agent, as defined in Eq. 6.

$$A_t = \begin{cases} \underset{A}{max}\ Q_t(S_t, A) & x < \epsilon \\ RandomChoice(A) & \text{Otherwise} \end{cases} \quad (6)$$

where x is a is a random value ranging within $[0, 1]$ that defines the agent's strategy for exploiting or exploring, while ϵ decreasing over time as Eq. 7:

$$\epsilon = \begin{cases} \epsilon_{max} - (steps \times decay) & \epsilon \geq \epsilon_{min} \\ \epsilon_{min} & \text{Otherwise} \end{cases} \quad (7)$$

where $decay$ is the decay rate parameter and identifies the amount of ϵ decrease after each step.

Next, the agent passes A_t to the environment, and after executing the prepared S_t, A_t pair in the environment, the agent receives the reward feedback R_t based on Eq.2 and the environment transits to the next state S'_t.

Meanwhile, the agent forms the $< S_t, A_t, R_t, S'_t >$ tuple, namely the experience tuple, and stores it in the experience replay buffer. During training, the DQN uses mini-batches sampled from this buffer using an adaptive sampling strategy (switching between random and prioritized sampling) to update the Q-values. In prioritized experience replay, the probability of choosing experience tuples with a larger Temporal Difference (TD) error is higher. TD error is the difference between the main NN output and the target NN output and defined in Eq. 8 as follows:

$$\begin{aligned} TD_{err} &= |Q_{target}(S_t, A_t) - Q_{main}(S_t, A_t)|, \\ pr_{exp} &= (TD_{error} + \zeta)^{\alpha} \end{aligned} \tag{8}$$

We store a priority value for each experience with pr_{exp} by adding $\zeta = 0.01$ to TD_{err} to avoid it becoming zero and powered by α which controls how much prioritization is required. For example, $\alpha = 0$ means that the agent behaves as random sampling. $Q_{target}(S_t, A_t)$ is calculated in Eq.9 as follows:

$$Q_{target}(S_t, A_t) = R_t + (1 - ter)\gamma \max_{A'} Q_{target}(S_{t'}, A') \tag{9}$$

where γ is the discount factor and ter is terminal state and indicates that the episode is over. During the training process, the loss function minimizes the square TD_{error}. It is also noteworthy that the weights of main NN are copied to target NN after every $target_{update}$ number of steps.

Although SA-DQN works well in optimizing routing decisions in our system model, it has two key limitations. First, it struggles to handle scenarios where multiple traffic flows arrive simultaneously, as the single-agent design cannot manage concurrent routing decisions efficiently. Second, when routing paths are updated even for existing active flows, it reveals opportunities for further improvements in network performance and resource utilization.

E. Cooperative MARL Routing

In the MARINA framework, while all DRL agents, each of which is responsible for one traffic flow routing decision, cooperate to optimize routing, a key aspect is the division of responsibilities among the agents. Specifically, one designated agent is still responsible for finding a routing path for new incoming traffic flow. The remaining agents are responsible for rerouting previously active flows, monitoring their status and updating paths as needed to adapt to dynamic network conditions such as changes in traffic, congestion, or QoS requirements. In MARINA, multiple DRL agents cooperate with each other to update routing decisions in a distributed manner while being trained centrally using the shared prioritized experience replay buffer as illustrated in Fig. 2. In this paper, the Centralized Training and Decentralized Execution (CTDE) paradigm [17] enables our DRL agents to access global network metrics and the status of traffic flows during the training phase. In the execution phase, each agent operates independently to reroute an existing traffic flow in the network, as well as an agent that routes the newly arrived traffic flow.

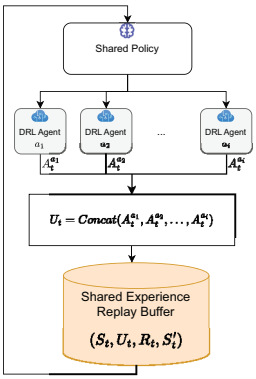


Fig. 2: Shared Prioritized Experience Replay Buffer in MARINA

As Fig. 2 illustrates, all existing agents in the environment store their own experience combined with all other agents in a shared prioritized experience replay buffer. In order to avoid confusion, we used U_t notation for concatenated actions at time t. MARINA shows improvements in terms of efficiently utilizing the underused links and satisfying QoS requirements of more traffic flows. Agents analyse the global network state and identify opportunities for either maintaining the current path or assigning a new path to the traffic flows. This cooperative approach enhances the adaptability of the routing algorithm, enabling it to handle varying traffic patterns and unexpected network changes while minimizing disruptions and maintaining QoS for all flows.

V. Performance Analysis

In this paper, we compare the performance of MARINA with other approaches including single-agent DQN, OSPF, and ECMP [1]. In our experiments, we initially analyzed the convergence of MARINA's reward values in the training phase and then evaluated the performance of four algorithms in terms of overall network links utilization and percentage of flows that meet QoS requirements in terms of throughput, latency, and combined metrics.

A. Experiments Setup

The experiments were carried out on a server equipped with 64GB of RAM, a 12th generation Intel Core i9 processor, and an NVIDIA RTX A5000 GPU. The operating system was Ubuntu 20.04 LTS. Python version 3.10 was used for developing our scripts. Furthermore, we used Numpy version 2.0.2, and PyTorch version 2.5.0 for developing the DRL agent employed on our simulated SDN controller. For the simulation environment, we used BNNetSimulator [6], a packet-level network simulator built on the OMNeT++ framework, to interact with the DRL agent. We set DRL agents' hyperparameters according to [3].

B. Topology and Traffic Flows Specifications

To evaluate the performance of our proposed method, a comprehensive experimental setup was designed, focusing on realistic network conditions and topology. The simulations were performed on the real-world GEANT2 topology from the TopoHub repository [9] commonly used in SDN research [3], [11]. The GEANT2 topology consists of 24 nodes and 38 undirected links connecting various pairs of nodes.

In this topology, each link has a 3 Gbps bandwidth capacity. The queue length of each node is equal to 32 packets inspired from [5]. The ingress-egress pairs for this topology are selected randomly. The nodes are interconnected in a way that includes multiple paths between source and destination pairs, allowing for dynamic and adaptive routing decisions. We consider three different traffic flow characteristics, which are aligned with 3 different network slices (eMBB, URLLC, and mMTC) commonly used in 5G networks. These traffic types are selected uniformly, and the order of these requirements, stringent to lenient QoS requirements, is inspired from [5]. In this setting, traffic type 1, representing eMBB traffic, requires 80 Mbps bandwidth, and tolerates up to 50 ms latency and a 0.25% packet loss ratio. Similarly, these characteristics, respectively, for type 2 and type 3 traffic flows, characterizing uRLLC and mMTC classes, are 40 mbps and 20 mbps for bandwidth, up to 15 ms and 1 s for latency tolerance, and up to 0.1% and 1% in terms of packet loss ratio.

In our experiments, these flow types are randomly chosen with a uniform distribution. Moreover, to evaluate the performance of the proposed algorithm under varying network loads, the lifetime of traffic flows are specified based on an exponential distribution with λ rates set to 75 and 100 steps. These different λ values represent how long traffic flows remain in the network and reflect varying levels of network traffic intensity, enabling us to simulate and analyze the algorithm's behavior under light and heavy load conditions. This approach ensures a comprehensive assessment of the algorithm's adaptability and efficiency in diverse traffic scenarios. Also, in order to ensure reliability of the results, each reported result is an average of 20 independent runs.

C. Experiments Results

In this section, results of various experiments are reported. We initially analyzed the effect of prioritized experience replay buffer on the convergence time of the DRL agent. Fig. 3 illustrates the DRL agent convergence using random sampling experience replay buffer ($\alpha = 0$) and prioritized experience replay buffer ($\alpha = 1$) over the designed reward function.

The convergence rate is a critical factor in reinforcement learning, as it determines how quickly the agent learns an effective policy for optimizing routing decisions. The average reward values stabilize after around 1000 episodes, indicating that the agent has effectively learned to balance QoS requirements and network resource usage.

Fig. 3 illustrates the MARINA agent's reward convergence over 2000 training episodes. Each point is a moving average of 20 episodes, and the reported value for each episode is the average of all the rewards gained at the end of the episode. We consider the number of steps per episode equal to 200 to make sure the lifecycle of traffic flows is properly simulated in the training environment.

As Fig. 3 depicts, the prioritized experience replay buffer ($\alpha = 1$) significantly accelerates the convergence in comparison with the random sampling experience replay buffer ($\alpha = 0$) strategy. This is because prioritized sampling ensures the agent focuses on high-impact transitions, improving learning efficiency. However, both approaches stablize after 1000 episodes indicating that the agent has effectively learned an optimal policy.

Fig. 4 highlights the superiority of MARINA over single-agent DQN (SA-DQN), ECMP, and OSPF in terms of overall network links utilization. The utilization of the network links for both $\lambda = 75$ and $\lambda = 100$ shows how efficiently traffic flows are distributed throughout the network. MARINA consistently achieves higher link utilization by dynamically spreading traffic across underutilized links, avoiding congestion on static paths commonly seen with OSPF and ECMP.

As depicted in Fig. 4, OSPF and ECMP fail to distribute the load across the network due to inadaptibility to dynamic traffic conditions, with OSPF yielding the lowest utilization due to its reliance on shortest paths and ECMP achieving moderate improvement through equal-cost path splitting. Although SA-DQN approach adapts to the dynamic nature of incoming traffic flows, it focuses solely on new traffic and neglects existing active flows. In contrast, MARINA's cooperative multi-agent framework enhances adaptability by efficiently balancing traffic across underutilized links while meeting QoS requirements. The low overall utilization values, such as 20% and 30%, are attributed to the smaller queue sizes in the network, which limit the amount of traffic that can be buffered and transmitted through the links.

Table I compares the percentage of flows that meet QoS requirements achieved by four routing algorithms—OSPF, ECMP, SA-DQN, and MARINA—under two traffic loads in terms of throughput, latency, and combined metrics. Throughput reflects the impact of the packet loss ratio, indicating

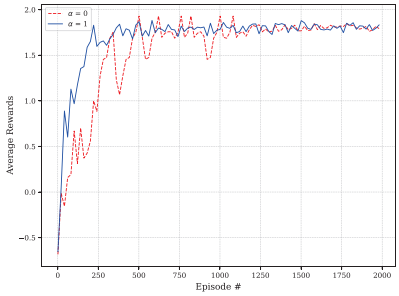
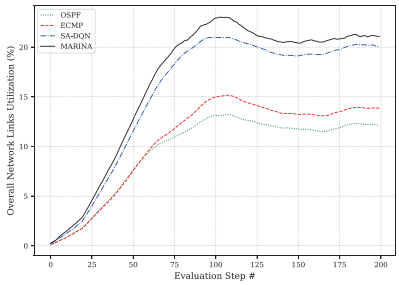
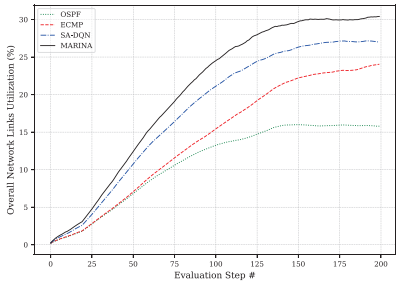


Fig. 3: MARINA agent training (moving average of 20 episodes)



(a) λ = 75



(b) λ = 100

Fig. 4: Overall Links Utilization ($\lambda = 75$, and $\lambda = 100$)

TABLE I: Percentage of flows that meet QoS requirements in terms of throughput, latency, and combined metrics.

Algorithm	Throughput (%)		Latency (%)		Combined (%)	
	$\lambda = 75$	$\lambda = 100$	$\lambda = 75$	$\lambda = 100$	$\lambda = 75$	$\lambda = 100$
OSPF	76.92	62.43	74.28	70.33	56.58	44.50
ECMP	77.65	69.20	82.50	79.92	62.95	55.33
SA-DQN	80.48	75.03	88.38	81.00	71.85	65.92
MARINA	83.50	78.13	90.10	84.50	80.78	72.23

the percentage deducted from the requested bandwidth while still meeting acceptable QoS requirements and ensuring the service remains unaffected by packet loss in the network. MARINA consistently achieves the highest performance, meeting 83.50% throughput, 90.10% latency, and 80.78% combined QoS at $\lambda = 75$, and maintains strong results under higher load ($\lambda = 100$). SA-DQN follows closely outperforming OSPF and ECMP. In contrast, OSPF shows the lowest QoS satisfaction, particularly under high load, highlighting the effectiveness of MARINA and SA-DQN in dynamic and congested networks.

VI. Conclusion

In this paper, we have introduced MARINA, a multi-agent RL-based QoS-aware routing algorithm for SDN that combines multi-agent DQN with shared prioritized experience replay buffer techniques. Our approach effectively addresses the limitations of existing routing methods by providing a more adaptive and comprehensive solution to managing multiple QoS constraints, including throughput, and latency, and efficiently utilizing network resources. The experimental results on GEANT2 real-world network topology demonstrate that MARINA consistently outperforms traditional routing algorithms in key performance metrics. Additionally, MARINA improves network resource efficiency by intelligently selecting optimal paths, increasing the network's capacity to handle future traffic demands. Future work will focus on further refining the algorithm and employing Graph Neural Networks and exploring its applicability in diverse network scenarios to continue improving routing efficiency and QoS management.

References

[1] Sai Shreyas Bhavanasi, Lorenzo Pappone, and Flavio Esposito. Dealing with changes: Resilient routing via graph neural networks and multi-agent deep reinforcement learning. *IEEE Transactions on Network and Service Management*, 20(3):2283–2294, 2023.

[2] Daniela M Casas-Velasco, Oscar Mauricio Caicedo Rendon, and Nelson LS da Fonseca. Intelligent routing based on reinforcement learning for software-defined networking. *IEEE Transactions on Network and Service Management*, 18(1):870–881, 2020.

[3] Daniela M Casas-Velasco, Oscar Mauricio Caicedo Rendon, and Nelson LS da Fonseca. Drsir: A deep reinforcement learning approach for routing in software-defined networking. *IEEE Transactions on Network and Service Management*, 19(4):4807–4820, 2021.

[4] Meignanamoorthi Dhandapani, V Vetriselvi, and R Aishwarya. Coopai-route: Drl empowered multi-agent cooperative system for efficient qos-aware routing for network slicing in multi-domain sdn. *CMES-Computer Modeling in Engineering & Sciences*, 140(3), 2024.

[5] Miquel Farreras, Jordi Paillissé, Lluís Fàbrega, and Pere Vilà. Gnnet-slice: A gnn-based performance model to support network slicing in b5g networks. *Computer Communications*, page 108044, 2025.

[6] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-fermi: Network modeling with graph neural networks. *arXiv preprint arXiv:2212.12070*, 2022.

[7] Jochen W Guck, Amaury Van Bemten, Martin Reisslein, and Wolfgang Kellerer. Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation. *IEEE Communications Surveys & Tutorials*, 20(1):388–415, 2017.

[8] Yingya Guo, Zhiliang Wang, Xia Yin, Xingang Shi, and Jianping Wu. Traffic engineering in sdn/ospf hybrid network. In *2014 IEEE 22nd international conference on network protocols*, pages 563–568. IEEE, 2014.

[9] Piotr Jurkiewicz. Topohub: A repository of reference gabriel graph and real-world topologies for networking research. *SoftwareX*, 24:101540, 2023.

[10] Murat Karakus and Arjan Durresi. Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, 80:200–218, 2017.

[11] Gyungmin Kim, Yohan Kim, and Hyuk Lim. Deep reinforcement learning-based routing on software-defined networks. *IEEE Access*, 10:18121–18133, 2022.

[12] Justus Rischke, Peter Sossalla, Hani Salah, Frank HP Fitzek, and Martin Reisslein. Qr-sdn: Towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks. *IEEE Access*, 8:174773–174791, 2020.

[13] Raheleh Samadi, Amin Nazari, and Jochen Seitz. Intelligent energy-aware routing protocol in mobile iot networks based on sdn. *IEEE Transactions on Green Communications and Networking*, 2023.

[14] Lizeth Patricia Aguirre Sanchez, Yao Shen, and Minyi Guo. Dqs: A qos-driven routing optimization approach in sdn using deep reinforcement learning. *Journal of Parallel and Distributed Computing*, 188:104851, 2024.

[15] Tom Schaul. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[16] Ning Wang, Kin Hon Ho, George Pavlou, and Michael Howarth. An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys & Tutorials*, 10(1):36–56, 2008.

[17] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song. Is centralized training with decentralized execution framework centralized enough for marl? *arXiv preprint arXiv:2305.17352*, 2023.

[18] Jie Zhu, Fengge Wu, and Junsuo Zhao. An overview of the action space for deep reinforcement learning. In *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, pages 1–10, 2021.