SDN-based Network Traffic Classification using Deep Reinforcement Learning

Sifeddine Salmi*, Miloud Bagaa*, Messaoud Ahmed Ouameur *, Oussama Bekkouche§ and Adlen Ksentini¶
* Université du Québec à Trois-Rivières, Trois-Rivières, QC, Canada.
Emails: {sifeddine.salmi, miloud.bagaa, messaoud.ahmed.ouameur}@uqtr.ca
, §Aalto University, Otakaari 24, 02150 Espoo FINLAND (e-mail: oussama.bekkouche@aalto.fi),
¶EURECOM, Campus SophiaTech, France (e-mail: adlen.ksentini@eurecom.fr)

***Abstract*—Software-Defined Networking (SDN) has emerged as a transformative technology that revolutionizes network management and architecture by providing unparalleled flexibility and control over data traffic flows. This flexibility is increasingly crucial in managing the complex demands of modern networks, whereby efficient traffic management is essential for mitigating congestion and enhancing operational efficiency. This paper introduces a novel traffic management model that employs Deep Reinforcement Learning (DRL) to transcend the conventional limitations typically associated with routing strategies that prioritize the shortest path or make non-optimal decisions when forwarding the traffic between different peers. Our model not only reduces overall network congestion but also aims to minimize bandwidth usage and enhance routing mechanisms within SDN environments. By incorporating DRL-based load balancing mechanisms, the model intelligently redistributes traffic across multiple pathways, shifting the focus from proximity to efficiency. This strategic redistribution prioritizes routes that optimize both, transmission time and network performance, rather than merely the shortest path. Moreover, the integration of DRL allows for real-time decision-making, enabling our system to dynamically adapt to changing traffic conditions and user demands. This capability is instrumental in significantly reducing transmission times and improving the overall efficiency of traffic flow across the network. Our findings highlight the substantial benefits of integrating SDN with advanced DRL techniques, offering a pioneering perspective on traffic routing within SDN networks. We evaluated the proposed framework via simulations and the obtained results demonstrated the efficiency of our solution compared to the baseline approaches.**

I. Introduction

In recent decades, technological advancements have catalyzed a paradigm shift in connectivity and communication. Notably, Software-Defined Networking (SDN) has emerged as a transformative technology, playing a pivotal role in driving this digital evolution. Unlike traditional networks, SDN-based networks undergo a profound metamorphosis, marked by centralized management facilitated by a dedicated controller. This distinctive feature endows SDN networks with a high degree of flexibility, making them a cornerstone in the contemporary era of dynamic and adaptive networking environments. Moreover, the SDN technology is playing a pivotal role in the cloud-edge continuum environment by efficiently interconnecting various micro-services and hosting nodes [1].

It has been noted that applications involving the exchange of substantial amounts of data can significantly affect the time required for traffic to navigate through the network. A lack of thorough comprehension of the network paths and their corresponding probabilities complicates the task of optimizing traffic flow. Numerous optimization solutions have been explored in existing literature to effectively direct traffic within SDN-based networks. However, the application of these optimization techniques often entails significant time delays before decisions are made [2]. This delay renders them less effective for emerging applications, which are characterized by heavy traffic loads and unpredictability of traffic patterns. To address this issue, a more refined and efficient approach is required to promptly respond to traffic changes and manage overloads.

In this paper, we propose an efficient framework that leverages the strengths of DRL and SDN technologies to make rapid decisions for directing traffic within an SDN-based network. The objective is to reroute traffic across multiple pathways to satisfy the emerging applications' demands for bandwidth, end-to-end delay, and jitter. By redistributing traffic across diversified paths, we aim to enhance overall network performance. This is achieved by considering various criteria, such as bandwidth, jitter, and delay on each pathway. We assume that the SDN controller has comprehensive information about the network topology, enabling it to make decisions for steering the traffic between any two hosts in the network using separate paths. The construction of a multi-path is beyond the scope of this paper. However, any algorithm in the literature, such as [3] can be employed for constructing the multi-path topology. Our solution focuses more on the routing protocol by enabling efficient steering of the network traffic via different paths to achieve the desired objectives.

Our proposed solution treats each SDN-enabled switch as an independent DRL agent, capable of making autonomous decisions regarding different data flows. These flows are classified based on characteristics, such as TCP and UDP ports and involved peers (i.e., source and destination). Subsequently, traffic is directed through various paths to satisfy the desired Service Level Agreement (SLA). At each switch, the incoming traffic flow from an ingress port is distributed across egress ports that facilitate access to the destination by leveraging already deployed multi-path topology. Additionally, each agent determines the volume of traffic to be sent through each egress port to achieve the intended objectives while avoiding network congestion. Through the continuous collection and

analysis of data from each path, including the estimation of expected transit times, we dynamically optimize traffic flow. This strategy not only boosts the overall efficiency of the network but also fosters a more adaptive and responsive communication infrastructure.

The rest of the paper is organized as follows: The next section is devoted to related works, whereby different related works are presented. Section III presents the problem formulation and the main idea of this paper. Meanwhile, section IV presents a detailed solution description. Last but not least, the simulation results are summarized in section V. Finally, the paper is concluded in section VI.

II. Related work

The authors in [4] have proposed an approach for optimizing the routing and resource management in an SDN-based network. The authors have proposed a model that favors either the shortest path or employs simplistic load-balancing strategies. In contrast to this work, our solution leverages DRL and new paradigms of SDN technology to steer the traffic efficiently. Our approach transcends the conventional limitations of routing strategies by prioritizing efficiency the bandwidth utilization to meet the desired SLA. By incorporating DRL-based load balancing mechanisms, our model intelligently redistributes traffic across multiple pathways, dynamically adapting to changing traffic conditions and user demands.

Meanwhile, Fu *et al* [5] undertake a study on SDN-enabled disjoint multi-path routing, showcasing the benefits of load-balancing over conventional shortest-path routing. In contrast to this work that propose a simple model for steering the traffic, our model is more advanced by integrating cutting-edge technologies, such as DRL technique, for real-time decision-making, thereby significantly enhancing the adaptability and efficiency of traffic management within SDN architectures.

Expanding upon this foundation, our review encompasses a spectrum of research endeavors elucidating the intricacies of multi-path routing. Notably, the works of Yan *et al* and [6] that propose innovative multi-path solutions leveraging OpenFlow queuing mechanisms. While their studies contribute to a deeper understanding of multi-path routing, our work introduces a pioneering perspective by integrating DRL techniques for dynamic traffic management, offering enhanced efficiency and adaptability in SDN networks. Moreover, our work leverages the new concepts Group tables offered by SDN technology for enforcing various decisions.

Further enriching our understanding is the exploration of adaptive multi-path provisioning schemes, exemplified by the dynamic path selection strategy proposed in [7]. While their approach focuses on maximizing bandwidth and availability, our model takes a holistic view, leveraging DRL to optimize traffic routing based on efficiency, bandwidth usage, and real-time network conditions.

Building upon these foundational studies, our research introduces a novel traffic management model that leverages DRL to transcend the limitations of conventional routing strategies within SDN environments. By prioritizing efficiency, adaptability, and real-time decision-making, our work offers a pioneering perspective on traffic routing, contributing significantly to the advancement of SDN network management via machine learning techniques.

III. Problem Formulation and Main Idea

In the context of related works, one notable aspect involves the development of Algorithms that enable the network to intelligently determine traffic routing through diverse paths. This strategic approach, central to optimizing temporal efficiency, aligns with our overarching goal of enhancing the user experience by providing the network with the capability to autonomously reshape traffic flows. To the best of our knowledge, we are the first to propose a DRL-based routing protocol for steering the traffic on a multi-path SDN-enabled network by leveraging the Group tables. To embark on this journey, we employed the ONOS REST API to meticulously gather device information from a controller. The proposed Algorithm focus on steering the traffic and orthogonal on any routing protocol that can be used to construct the network paths between different hosts. Before starting the execution of our framework, multi-path paths should be created using any existing tool, such as the one proposed in [3].

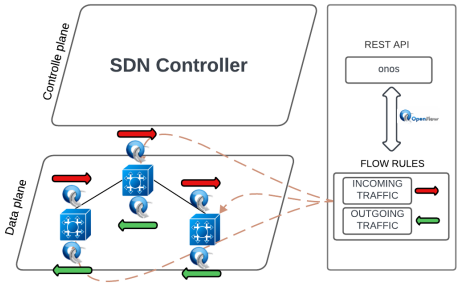

Fig. 1. Main idea and problem formulation

Fig. 1 depicts the main idea of our solution. which is executed in conjunction with the SDN controller. We have generated a closed-loop mechanism that consists of continuous monitoring and policy enforcement enabling Zero-touch network and Service Management (ZSM) [8]. For this reason, continuous monitoring of ongoing traffic at each network switch and host is carried out to check the conformance of network throughput to the SLA. Subsequently, according to the perceived QoS and SLA, new policies are enforced into the network to enhance the quality of decisions and adapt according to the changes happening in the network. The decisions enforce new flow rules that are pushed at each switch to steer the traffic to achieve better performance. The agent for each switch leverages DRL, more precisely Deep Deterministic Policy Gradient (DDPG), for making the right decisions that enhance the QoS and increase bandwidth utilization.

Once the agent determines the appropriate actions for each switch, this decision is manifested through the implementation of two distinct flow rules per switch for each traffic flow, as

illustrated in Fig. 1. For each active traffic flow, these two rules play pivotal roles in traffic management. The first rule is specifically designed to manage incoming traffic at each ingress port. This involves detailed inspection and filtering based on predefined criteria, ensuring that only the intended traffic is allowed to enter the network, thereby optimizing resource utilization and enhancing security measures.

The second rule governs the outgoing traffic at each egress port. It prioritizes traffic based on various parameters, such as destination, quality of service (QoS) requirements, and current network conditions. This rule is crucial for efficiently allocating bandwidth and preventing congestion, as it dynamically adjusts the flow based on real-time network performance data.

Together, these rules ensure that the network can handle traffic flows intelligently and efficiently. By segregating the control mechanisms for ingress and egress traffic, the system maintains high throughput and minimizes latency, thereby ensuring that network performance aligns with the strategic objectives of traffic management within SDN environments. This dual-rule approach not only simplifies the management of complex traffic scenarios but also enhances the robustness and responsiveness of the network infrastructure.

IV. Detailed Solution Description

Our study delves into the intricacies of group tables, examining their structure and functionality, particularly in the context of achieving effective load balancing within networks. Group tables play a crucial role in network management, guiding the flow of traffic to optimize performance. Understanding the intricacies of group tables is essential as they form the foundation for implementing efficient load-balancing strategies within networks. Load balancing (LB) is a fundamental aspect of network design and operation, ensuring optimal performance by distributing traffic across network devices [9]. LB is essential in both Software-Defined Networking (SDN) and cloud computing [10], guaranteeing continuous service delivery even under network congestion conditions or device failures [11]. Within SDN environments, LB dynamically allocates local workloads across networks or pathways [12]. Additionally, it prevents server overload while maximizing resource utilization.

A group table consists of group entries, each identified by a unique 32-bit integer. The capability for a flow entry to reference a group allows OpenFlow to incorporate additional forwarding methods, such as 'select' and 'all' [13]. Understanding how group tables operate is crucial for implementing effective load-balancing mechanisms within networks.

These entries encompass crucial elements:

- Group identifier.
- Group type, determining group semantics.
- Counters, updated during packet processing.
- Action buckets, an ordered list containing sets of actions and associated parameters.

Our primary objective is to enhance load-balancing algorithms, which can profoundly influence traffic distribution. Instead of selecting a single path, we aim to generate a more dynamic approach by efficiently splitting the traffic across multiple paths to ensure the required SLA in an efficient and timely manner. The Group Table already incorporates Weighted Selection with the SELECT forwarding method for optimal load-balancing. This integration prioritizes action buckets based on their weights, ensuring efficient traffic distribution across network paths by favoring higher-weighted buckets.

To provide a programmable solution and enhance adaptability, we leverage flow rules to dynamically adjust flows based on changes in the group table. This approach enables the system to respond effectively to evolving network conditions. Recognizing the increasing demand for adaptive load-balancing mechanisms within dynamic networking environments, we have integrated DRL techniques into our system. Contemporary networking landscapes necessitate load balancers to dynamically adjust traffic distribution in real-time, considering factors, such as congestion and link performance. Traditional load balancers excel in static scenarios but often struggle to adapt under changing conditions. By integrating DRL, our system autonomously learns and it will be able to adapt optimally according to different traffic patterns and users' behaviors.

A. Implementation Details

The integration of DRL aims to empower the load-balancing framework with the capability to dynamically adjust traffic distribution policies in response to evolving network conditions. This adaptation is essential for ensuring efficient resource utilization and minimizing network latency, particularly in SDN-based networking environments characterized by their dynamic and adaptable nature. The proposed framework can be modeled as a model-free Markov Decision Process (MDP) problem, defined as a tuple $\langle S, A, R \rangle$, such that: $i)$ S denotes the set of states; $ii)$ A denotes the set of actions; $iii)$ R denotes the reward.

We consider the state set $S = \{s_1, s_2, \ldots, s_n\}$ to be a discrete states. We formulate each state s_t at a step t to consider the traffic delay T_t computed as the difference between time generation and time reception of each flow. The state also considers the total amount of the data processed across all buckets P_t. Moreover, the state considers the data rate represented as bandwidth B_t and the amount of data transferred D_t across all the flows. Thus, we represent the state S_t as: $S_t = (T_t, P_t, B_t, D_t)$.

Additionally, A is the set of possible actions that determine the weights allocated to buckets within the group table. In fact, each potential path between each source and destination at each switch is assigned a specific weight. The latter will be converted to the rate of the packet should be forwarded via that egress port.

Meanwhile, the reward $\mathcal{R}$ is defined as the average bandwidth between any source and destination in the network. This reward serves as a direct reflection of the effectiveness of the agent's actions. By prioritizing bandwidth optimization, the agent learns to adjust its actions to maximize bandwidth

performance. Consequently, this adaptation enhances network efficiency and overall performance, particularly in dynamic environments.

This decision-making process is crucial for orchestrating efficient resource allocation within the network. The resultant instructions are then transmitted to the data plane, whereby various tasks, such as forwarding and packet processing, are executed according to the taken actions by the DRL-based agent. Upon completing this process, the data plane provides performance results in the form of updated group tables. These outcomes essentially reflect the effectiveness of the actions derived from the decision-making process, and hence they are relayed back to the DRL-based agent. This feedback mechanism serves as a rewarding mechanism, reinforcing the agent's learning process and facilitating continual optimization of load-balancing strategies within the network architecture.

B. Training and Evaluation

After thoroughly defining our model-free Markov Decision Process (MDP) problem, it became evident that we face challenges associated with continuous state and action spaces, necessitating the use of the Deep Deterministic Policy Gradient (DDPG) algorithm. This algorithm offers a suitable framework for addressing our complex problem. Thanks to the Actor-critic strategy employed by DDPG, the agent will converge fast and accept continuous space for both the state and the action.

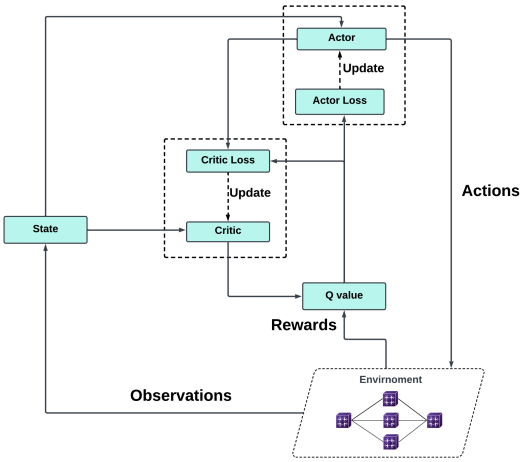

Fig. 2. DDPG-based Agent in the proposed solution

The learning process iterates through multiple episodes, allowing the DRL agent to engage with the environment, experiment with various strategies, and acquire optimal load-balancing policies. Initially, the agent adopts a randomized action selection strategy to explore the expansive action space and gain insights into the network dynamics. This phase of exploration is crucial for gathering essential information that will inform subsequent decision-making processes.

Strategy: Fig.2 illustrates the concept of strategy, often denoted as network knowledge, constitutes a fundamental aspect of reinforcement learning (RL) paradigms, particularly in the domain of decision-making. Formally, a policy π embodies this strategic approach by defining a probabilistic mapping from states to actions within the RL framework. Mathematically, this can be expressed as:

$$\pi_\theta(a_t|s_t) = \mathbb{P}[A_t = a_t|S_t = s_t] \tag{1}$$

Here, $\pi_\theta(a_t|s_t)$ represents the probability of taking action a_t given state s_t, parameterized by θ. Thus, the policy acts as the 'actor' in the RL system, determining the agent's actions based on the observed state.

We also have the Critic component, which provides estimates of Q-values $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$, whereby ϕ represents the parameters of the Critic. The Critic learns to estimate the value of state-action pairs, providing feedback on the effectiveness of actions selected by the Actor in achieving the agent's goals.

Simultaneously, the Critic, represented by $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$, evaluates the value of these actions in the given state. Through the Actor-Critic framework, the agent learns to improve its policy by receiving feedback from the Critic on the estimated value of state-action pairs.

However, the ultimate objective is to transition towards more deterministic actions as the agent learns and refines its policy. This transition is achieved through training both the Actor and the Critic.

We commence our endeavor by initiating the training of the Critic component, a foundational step in our pursuit of refining the agent's policy towards deterministic actions. The principal aim in training the Critic is the minimization of a designated loss function $L(\phi)$, where ϕ represents the parameters of the Critic network. This loss function quantifies the discrepancy between the predicted Q-values $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$ and the target Q-values $y^*(s, a)$. Typically, $L(\phi)$ is formulated as the squared error between the predicted and target Q-values:

$$L(\phi) = \left(y^*(s, a) - \hat{Q}_\phi^{\pi_\theta}(s_t, a_t)\right)^2 \tag{2}$$

Concurrently, for each sample i, the Temporal Difference (TD) error δ_t is computed, representing the deviation between the predicted Q-value at the current state-action pair (s_t, a_t) and the sum of the immediate reward r_t and the discounted estimated Q-value of the subsequent state (s_{t+1}) under the policy π_θ:

$$\delta_t = r_t + \gamma \hat{Q}_\phi^{\pi_\theta}(s_{t+1}, \pi_\theta(s_{t+1})) - \hat{Q}_\phi^{\pi_\theta}(s_t, a_t) \tag{3}$$

Subsequently, the target Q-values y_i are computed for a minibatch of N samples $\{s_i, a_i, r_{i+1}, s_{i+1}\}$. These values, signifying the TD target values for each sample, are calculated employing a target network Q' to mitigate issues associated with the moving target problem:

$$y_i = r_{i+1} + \gamma \hat{Q}'^{\pi_\theta}_{\phi'}(s_{i+1}, \pi(s_{i+1})) \tag{4}$$

Finally, the parameters ϕ of the Critic network are updated by minimizing the loss function L over the minibatch of samples:

$$L = \frac{1}{N} \sum_i (y_i - \hat{Q}_\phi^{\pi_\theta}(s_i, a_i|\phi))^2 \tag{5}$$

Through iterative minimization of the loss function utilizing environment samples, the Critic gradually approximates the true Q-values [14]. This approximation aids in evaluating the effectiveness of actions taken by the Actor according to the prevailing policy, thereby guiding the Actor toward making informed decisions within the environment.

Once the Critic has been trained and the target Q-values y_i have been calculated, the actor will be also trained using the same loss function. The deterministic policy gradient theorem (Silver et al., 2014) provides the basis for training the Actor. According to this theorem, the policy gradient is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t, a_t \sim \pi_\theta(\cdot)} \left[\nabla_a Q_\phi^{\pi_\theta}(s_t, a_t) \nabla_\theta \pi_\theta(s_t) \right] \quad (6)$$

This equation represents the guiding principle behind updating the policy parameters of the Actor network, θ, to improve the agent's decision-making process. Let's delve into the components of this equation to understand its implications for training the Actor.

- $\nabla_\theta J(\theta)$ represents the gradient of the objective function $J(\theta)$ with respect to the policy parameters θ. In simpler terms, it indicates how the expected cumulative rewards change with small changes in the policy parameters. This gradient serves as a compass guiding the Actor towards actions that maximize expected returns.
- The notation $s_t, a_t \sim \pi_\theta(\cdot)$ signifies that the states s_t and actions a_t are sampled from the current policy π_θ, which is parameterized by θ. This sampling process allows the Actor to explore different actions in different states, which is crucial for learning an effective policy.
- The policy function $\pi_\theta(s_t)$ maps states s_t to actions and represents the probability of selecting action a_t given state s_t. It encapsulates the decision-making process of the Actor, determining which actions to take in different states based on the learned policy parameters.
- The action-value function $Q_\phi^{\pi_\theta}(s_t, a_t)$ under policy π_θ, parameterized by ϕ, represents the expected cumulative rewards obtained by taking action a_t in state s_t and following policy π_θ thereafter. It serves as a measure of the long-term desirability of taking a particular action in a given state.
- Finally, the gradients $\nabla_\theta \pi_\theta(s_t)$ and $\nabla_a Q_\phi^{\pi_\theta}(s_t, a_t)$ denote the gradients of the policy function and action-value function, respectively. These gradients drive the learning process by indicating how changes in the policy parameters affect the action choices and subsequent rewards.

Incorporating the actor loss formulation, the actor loss L_{actor} can be expressed as:

$$L_{\text{actor}} = \frac{1}{N} \sum_i Q_\phi^{\pi_\theta}(s_i, a_i) \quad (7)$$

, where N represents the total number of samples (states) used in the calculation. This formulation highlights the objective of the Actor network to maximize the sum of Q-values obtained from the Critic network, as it strives to achieve higher expected returns/Q-values.

The policy gradient theorem provides a principled approach to updating the Actor's policy parameters based on the expected rewards obtained from different actions in various states. By iteratively adjusting the policy in the direction suggested by these gradients, the Actor strives to maximize expected returns, ultimately leading to improved decision-making capabilities in reinforcement learning tasks.

V. Performance Evaluation

In the balance of this section, we initially present the trainning of the DRL-based agent, then we present its inference.

We initiated an extensive simulation study to evaluate the effectiveness of our DDPG-basedload balancer agent. The primary objective was to assess its efficiency in balancing loads across interconnected components under various network conditions. To achieve this, we carefully designed a setup capable of generating three distinct traffic types, each with different data rates. This configuration allowed for a thorough evaluation of the load balancer's efficacy, including its performance under bandwidth restrictions and packet loss scenarios. Additionally, we enriched the environment by incorporating multiple paths, increasing the complexity of load-balancing assessments.

A. Training

For implementation, we utilized the PyTorch framework and conducted a rigorous training regime comprising over 4,000 episodes, each consisting of 50 epochs. To create our experimental environment, we employed the TCLink class within Mininet, enabling us to define unique criteria for each link. To initiate network traffic, we utilize the iperf tool, launching three concurrent traffic streams simultaneously. Within this framework, we established different distinct paths, each characterized by varying metrics to emulate real-world network scenarios.

Fig. 3. Running average of the previous 100 episodes using DDPG

To gauge the effectiveness of the DDPG policy, we monitor the running average reward computed over the preceding 100 episodes. This ongoing assessment allows us to evaluate the agent's performance and its capacity to adjust its policy in response to environmental cues.

Moreover, our analysis encompasses the examination of the running average reward, as illustrated in Fig. 3. This scrutiny

not only showcases the reward's convergence over time but also offers valuable insights into the DDPG agent's learning trajectory.

B. Inference

We have compared our solution to a base-line approach that distributes the weights equally across all the buckets. The comparative results, as depicted in Fig. 4, highlight the superior performance of the DDPG-based load balancer over the central Baseline solution, particularly in efficiently handling data packets as data rates increase. For instance, for a data $120Mbps$ rate, while the baseline solution did not exceed 4600 packets, our approach reaches 7500, which demonstrates its efficiency.

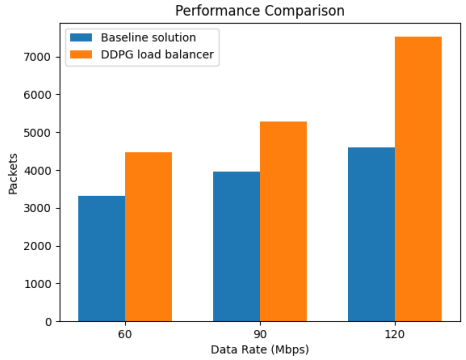

Fig. 4. Performance Comparison of Load Balancer Systems

Moreover, notably, the DDPG-based Load Balancer consistently outperforms the baseline solution, demonstrating substantial enhancements in bandwidth, as depicted in Fig.5. This persistent trend of bandwidth improvement not only demonstrates the effectiveness of our approach but also accentuates its reliable and consistent ability to achieve higher bandwidth gains. These discoveries underscore the superiority of the DDPG-based Load Balancer agent in optimizing data transmission performance.

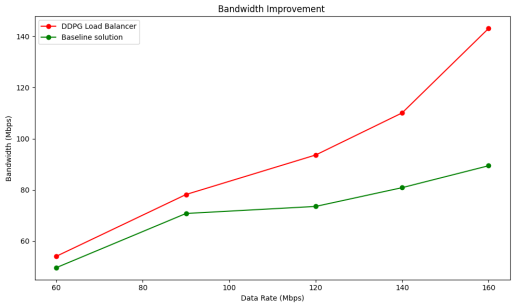

Fig. 5. Analyzing Bandwidth Performance: A Comparative Study of DDPG-based Load Balancers agent and baseline solution

Overall, our simulation outcomes illustrate the potential of our approach in enhancing network performance. It offers an optimized decision-making process capable of adapting to the dynamic nature of data plane environments and hence achieving the desired SLA whatever the traffic patterns or user's behavior.

VI. Conclusion

Our study demonstrated the effectiveness of a machine learning-based agent, more precisely DDPG, for enabling load balancing in optimizing traffic management within SDN. In our simulation, we directly compared the DDPG-based load balancer agent with a baseline solution, revealing significant enhancements in network performance with the DDPG approach. Notably, the DDPG-based agent load balancer showcased remarkable adaptability to dynamic network conditions, consistently achieving superior bandwidth utilization compared to the baseline. This underscores the immense potential of reinforcement learning algorithms, such as DDPG, in elevating traffic routing efficiency within SDN architectures. Looking ahead, future research endeavors could concentrate on further refining the DDPG algorithm to dynamically adapt to evolving network conditions and traffic patterns, thus fostering the development of more sophisticated load-balancing mechanisms. Moreover, the evaluation of our approach on SDN-based physical switches.

References

[1] N. Toumi, M. Bagaa, and A. Ksentini, "Machine learning for service migration: A survey," *IEEE Communications Surveys Tutorials*, vol. 25, no. 3, pp. 1991–2020, 2023.

[2] A. Nacef, M. Bagaa, Y. Aklouf, A. Kaci, D. L. C. Dutra, and A. Ksentini, "Self-optimized network: When machine learning meets optimization," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[3] M. Bagaa, D. L. C. Dutra, T. Taleb, and K. Samdanis, "On sdn-driven network optimization and qos aware routing using multiple paths," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4700–4714, 2020.

[4] M. R. Celenlioglu and H. A. Mantar, "An SDN based intra-domain routing and resource management model," *Proc. IEEE Int. Conf. Cloud Eng.*, pp. 347–352, Mar. 2015.

[5] M. Fu and F. Wu, "Investigation of multipath routing algorithms in software defined networking," in *Proc. Int. Conf. Green Informat. (ICGI)*, Aug. 2017, pp. 269–273.

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

[7] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Adaptive and reliable multipath provisioning for media transfer in SDN-based overlay networks," *Comput. Commun.*, vol. 106, pp. 107–116, Jul. 2017.

[8] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "Qos and resource-aware security orchestration and life cycle management," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2978–2993, 2022.

[9] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening geographical load balancing," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 657–671, Apr 2015.

[10] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009. [Online]. Available: http://dx.doi.org/10.1109/ncm.2009.218

[11] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems - SIGMETRICS '11*, 2011. [Online]. Available: http://dx.doi.org/10.1145/1993744.1993756

[12] V. D. Chakravarthy and B. Amutha, "Path based load balancing for data center networks using sdn," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, p. 3279, Aug 2019.

[13] "Openflow switch specification," https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.4.1.pdf, march 26, 2015.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*.