Load Balancing Optimization in Software-Defined Wide Area Networking (SD-WAN) using Deep Reinforcement Learning

Mohamed Amine Ouamri, Gordana Barb, Daljeet Singh, Florin Alexa

Load Balancing Optimization in Software-Defined Wide Area Networking (SD-WAN) using Deep Reinforcement Learning

Conference Paper · November 2022

DOI: 10.1109/ISETC56213.2022.10010335

4 authors:

Mohamed Amine Ouamri
École Nationale Supérieure d'Informatique et de Mathématiques
20 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Gordana Barb
Polytechnic University of Timisoara
18 PUBLICATIONS **76** CITATIONS
SEE PROFILE

Daljeet Singh
University of Oulu
27 PUBLICATIONS **96** CITATIONS
SEE PROFILE

Florin Alexa
Polithnica University Timisoara
68 PUBLICATIONS **145** CITATIONS
SEE PROFILE

Load Balancing Optimization in Software-Defined Wide Area Networking (SD-WAN) using Deep Reinforcement Learning

Mohamed Amine Ouamri[1], Gordana Barb[2], Daljeet Singh[3], Florin Alexa[2]

[1] Computer Communication, Grenoble Alpes university, LIG Laboratory, DRAKKAR Teams, Grenoble, France;
[2] Faculty of Electronics, Telecommunication and Information Technologies, Politehnica University Timisoara, Timisoara, Romania;
[3] Centre for Space Research, Department of Research and Development, School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, India;
ouamrimouhamedamine@gmail.com, daljeetsingh.thapar@gmail.com, {gordana.barb and florin.alexa}@upt.ro

2022 International Symposium on Electronics and Telecommunications (ISETC) | 978-1-6654-5150-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISETC56213.2022.10010335

***Abstract*— Software-Defined Wide Area Network (SD-WAN) holds tremendous potential to provide multi-cloud multi-network interconnection and prevent channel congestion. However, traffic among Customer Premises Edge (CPE) and controllers continuously increases, requiring pre-emptive load balancing in the control plane. In this paper, we study the flow migration problem in SD-WANs when the controller presents a limited processing capacity. Specifically, the data plane may include one or more CPE deployed at a site where service traffic is forwarded. To address this narrow, we propose a new approach based on a Deep Reinforcement Learning (DRL) strategy to optimize the balancing process under a latency constraint. As far as we can tell, we have not observed any pertinent research published in this context. The obtained simulation results revealed that our proposed approach decreases the load balancing and outperforms other baseline methods.**

***Index Terms*— SD-WAN, CPE, Load Balancing, artificial intelligence, MADQN.**

I. Introduction

Today, software-defined wide area networking (SD-WAN) has emerged as a popular technology that meets the connectivity requirements of different services and provides an effective quality of experience (QoE) [1]. Owing to their specific characteristics in comparison to wide area network (WAN), SD-WAN solutions logically combine several WAN links for increased capacity [2]. Moreover, SD-WAN can support WAN in a variety of scenario such as traffic management. For example, when the link is congested or deteriorated between two sites, traffic is re-routed to other available and less congested links [3]. The performance of the SD-WAN depends on the controller's capacity and communication protocol. Generally, the controller automatically establishes a connection with several customer premises edge (CPE) which are deployed at different sites. Specifically, CPE has no direct knowledge of the network infrastructure and must be able to provide diverse access capabilities to accommodate different WAN access environments [4]. In large-traffic SD-WAN, controllers must process and respond as quickly as possible to the huge number of services. However, controllers are usually resource constrained [5] and can be overloaded due to the dynamic flow. Therefore, a load imbalance is observed in the control plan [6]. Data plane flow migration is an important and widely used method for load balancing between controllers, which should be fully understood. It consists in redirecting the flow from a particular CPE to other underloaded controllers [7]. Although SD-WAN has been studied in few articles, e.g., [8] [9], no article addressed the issue of load balancing in the scenario where SD-WAN sites are equipped by CPEs. For instance, authors in [9] proposed a deep reinforcement learning (DRL) to overcome the load in the WAN channels between headquarter (HQ) and branch site. The main objective is to improve the service availability, by migrating the flow between two channels. However, the analysis in this work does not take into account the capacity of the controller and use only broadband internet as external network. In [10], the authors introduced a load prediction based alertness approach to reduce the burden of the controllers. The proposed solution satisfies the requirements of the LAN and WAN controllers separately. The suggested architecture does not include CPEs and assumes a global knowledge of the infrastructure. In this paper, we focus on component migration to balance the load between controllers for SD-WAN where HQ and branch site use the insert CPE. We first proposed the SD-WAN architecture based internet broadband and multi-protocol label switching (MPLS). Afterwards, we model our problem as a non-linear binary program to jointly solve our optimization for minimizing the load balancing and running costs of this operation. In this regard, we adopt multi agent deep reinforcement learning (MADRL) to automatically identify the controller load and direct the flow to less saturated controllers.

II. System Model And Problem Formulation

In the following, we present a simplified SD-WAN

978-1-6654-5150-5/22/$31.00 ©2022 IEEE

architecture and formulate our problem. The research objective is to develop a computationally simple model to minimize both load balancing and cost migration.

A. System Model

We consider a SD-WAN comprising one-branch site and two (HQ) interconnected through two alternative cloud network such as internet and MPLS. According to network function [4], SD-WAN controllers are deployed in control plane and manages the entire data plan. Moreover, the data transfer between the data plane and control plane is performed via southbound interface as shown in fig.1. Without loss of generality, a network $G(\xi, \Omega)$ is assumed, where ξ designates the set of CPEs and Ω denotes the set of links between CPEs $\alpha_i = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ and controllers $C_j = \{C_1, C_2, \dots, C_m\}$. It is worth mentioning that each controller j is characterised by limited processing capacity $\omega_j = \{\omega_1, \omega_2, \dots, \omega_m\}$, with m is the total number of controllers and each link i has latency δ_i. For any pair of CPE, the supervision is carried out by a single controller. However, we assume east-west communication to connect the controllers and schedule flow migrations. Let's define a binary variable x_{ij} that indicates the association between CPEs and controllers, where $x_{ij} = 1$ if and only if CPE i is supervised by controller j, otherwise 0. As discussed above, the load of a controller is related to the number of flows received from CPEs. Therefore, the average load of a particular controller can be expressed as [7]

$$L_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}\,\lambda_i \tag{1}$$

where λ_i represents the number of flow requests sent by CPE.

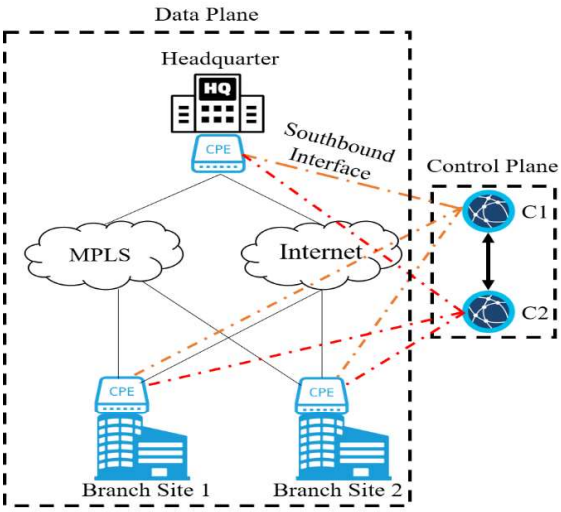


Fig.1. Distributed SD-WAN architecture.

To avoid an overload problem in control plane, we introduce the response time of controller as a metric to address the load-balancing problem. In literature, it is common that the response time of a controller depends on its processing capacity ω_j and in accordance with its load L_j. Thus, by applying queuing theory, the SD-WAN controller can be modelled as $M/M/1$ and response time is given by

$$\Gamma_{C_j} = \frac{1}{\omega_j - L_j} \tag{2}$$

Maintaining adaptive and continuous fairness between controllers in terms of response time, requires load balancing in the control panel. In fact, controller could be overwhelmed when dealing with many transactions, i.e. the incoming flow exceeds the processing capacity $\lambda_i >> \omega_j$ [6]. Therefore, migrating the flow to an annoying controller (see fig. 1) remains significant. In this work the load-balancing index can be calculated as

$$LB = \sum_{j=1}^{m}\sum_{j'>j}^{m} \left|\Gamma_{C_j} - \Gamma_{C_{j'}}\right| \tag{3}$$

where $\Gamma_{C_{j'}}$ is the response time of an adjacent controller. It is clear from the expression above that the smaller the difference $\left|\Gamma_{C_j} - \Gamma_{C_{j'}}\right|$, the stronger is the load balancing. In this work, the reliability of CPEs has not been taken into consideration and may be a failing factor for load balancing.

B. Problem Formulation

This section introduces the problem formulation employed to optimize load balancing in case the controller is saturated and unable to process incoming flows, taking into account the cost migration. To solve the issue in an optimal way, we formulate our first objective as a load balancing minimization problem. Mathematically, this can be expressed as

$$\min_{\omega_j, \delta_i, L_j, \Gamma_{C_j}} LB \tag{4}$$

s.t.

$$\mathbf{C1}: \sum_{\alpha\in\Omega} \delta_i x_{ij} < \rho, \qquad \forall\, i,j \in \xi \tag{5}$$

$$\mathbf{C2}: \sum_{j=1}^{m} x_{ij} = 1, \qquad \forall\, i,j \in \xi \tag{6}$$

$$\text{C3}: \; L_j < \omega_j, \qquad \forall\, j \in \{1,2,\dots,m\} \tag{7}$$

$$\text{C4}: \; \Gamma_{C_j} < \Gamma^{th}, \qquad \forall\, j \in \{1,2,\dots,m\} \tag{8}$$

Constraint **C1** indicates that CPE-controller communication latency should not exceed a certain threshold ρ. Constraint in **C2** means that each CPE is associated with only one controller. **C3** and **C4** impose that the load of controller must not be allowed to exceed its limited processing capacity and the response time of controller cannot exceed a specified threshold. Flow migration between controllers is realized by exchanging several messages and to notify each other. Consequently, achieving load balancing is costly. As illustrated in fig.2, OpenFlow-based migration operation is performed in two phases. First phase consists of initiating the migration process with a target controller whereas the second phase focuses on making the target controller like a master controller. Using this

process, we assume that the overloaded controller needs four messages to start the migration (Phase 1) and the target controller also requires four messages to accept the flow from CPE (Phase 2). Accordingly, we define migration costs as [4]

$$\Im_{C_j} = \left(\sum_{j=1}^{m} \left(4 \times \Gamma_{C_j} \times x_{ij} \right) + \sum_{j'=1}^{m} \left(4 \times \Gamma_{C_{j'}} \times x_{ij'} \right) \right) \times \Delta (9)$$

where Δ designates the operation cost. Our optimization is based on reducing the total migration cost. Hence, the problem can be formulated as follows

$$\varphi_{C_j} = min \sum_{j=1}^{m} \Im_{C_j} \quad (10)$$

s.t. **C1**, **C2, C3** and **C4**.

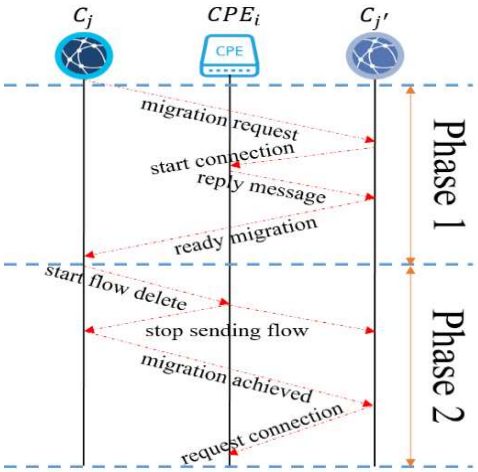


Fig.2. Messages exchanged for the migration process.

Both objectives given in equations (4) and (10) can be added together, thus transforming the problem into a weighted sum as

$$min\, w_1 LB + w_2 \Im_{C_j} \quad (11)$$

where $\sum w = 1$ is the weight factor.

III. Deep Reinforcement Learning

Deep reinforcement learning has received significant attention to enhance networks performance, where agent interact in the environment to achieve suboptimal solutions based on its actions. Indeed, using an optimal decision policy, the agent observes the state of the network (environment) and executes an action. The observation may include the full state information or only a partial state of the environment. Finally, the agent receives a reward or a penalty depending on the change in the state of the environment. Among the DRL algorithms applied to the load balancing problem, deep Q network (DQN) algorithm remains the most widely introduced approach in the literature [11] [12]. Nevertheless, it suffers from several disadvantages such as slowness of learning process caused by the number of transitions involved in learning a policy, but also environmental interaction. Moreover, DQN is limited in its ability to learn continuous actions [9]. Recently, deep deterministic reinforcement learning (DDRL) is suggested to meet the different constraints of the DQN. DDRL combine DQN with deterministic policy gradient (DPG) to generate policy function and Q-function. As shown in fig.4, the strategy and the Q-value are generated by two different architectures [13]. Actor architecture applies the DPG method to determine the strategy, whereas critic architecture adopts the DQN approach to determine the Q-value. Both architectures include neural network for learning and training (Online network), and the other (Target network) to disrupt the correlation of training data. Elsewhere, the transition information from each interaction with the online network is stored in an experience replay memory [12], which will be exploited later as a training mini batch for neural networks [13].

In the training process, the target actor/critic network is updated to reinforce the sustainability of the process. This step consists on the one hand to estimate the policy gradient at the critical network level after an action assessment by means of a value function. The new gradient estimate is forwarded to the actor network in order to obtain an optimal action. On the other hand, the neural network parameters are updated for actor module. It is worth noting that the update is achieved through a slow change, to maintain reliability. Here, the policy gradient is computed as [14]

$$\nabla_{\theta^\mu} J = grad[Q] * grad[\mu] \quad (12)$$

$$\approx \frac{1}{N} \sum\nolimits_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i} \quad (13)$$

where $grad[Q]$ and $grad[\mu]$ are the action gradient and parameter gradient from critic and actor network, respectively, $\mu(s_i)$ represents the action strategy calculated by the algorithm. To undertake the indispensable updates in critical losses, these are minimized by using the following function [15]

$$L = \frac{1}{N} \sum\nolimits_i (y_i - Q(s_i, a_i)|\theta^Q)^2 \quad (14)$$

where s and a are the state and action, respectively, and y_i designates the Q-value of the target network which is given as

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \quad (15)$$

In the above formula, $\gamma \in [0\ 1]$ is the discount factor used to actualize the future reward r_i, and s_{i+1} is the next state. In addition, y_i is obtained by performing the next action a_{i+1}based on μ'. The pseudo code for DDPG is outlined in the algorithm 1 [16]. The model we propose assumes that SD-WAN controller acts as an agent and makes decisions based on important information such as the response time, load and capacity processing of controllers. Following, we define state spaces, action spaces and reward function for DDPG.

State and Observation: At each time t, we consider the observed state by SD-WAN controller (agent) defined as $s_t = \{\Gamma_{C_j}, \omega_j, L_j\}$, where Γ_{C_j} represents the response time and ω_j , L_j

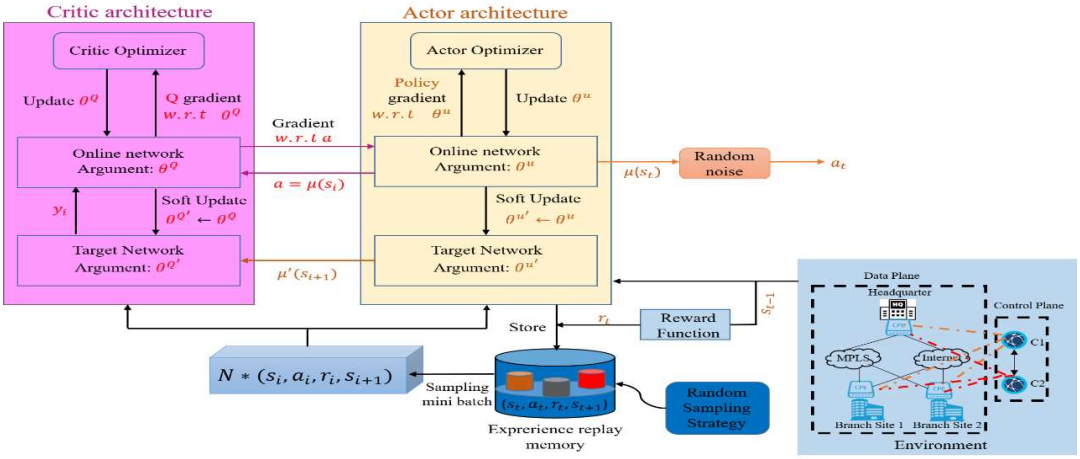


Fig.3. Deterministic deep policy gradient algorithm process.

are capacities and load of each controller j, respectively.

Action: At time step $t \in T$, SD-WAN controller takes an action space a_t by migrating the flow to another underloaded controller. The action space of SD-WAN controller is denoted as $\mathcal{A}_i = \{C_1, C_2, \dots, C_m\}$ which is defined by the set of typical controllers that can receive the flow, except the overloaded controller. The action taken by agent is achieved if and only if $L_j < \omega_j$ and $\Gamma_{C_j} < \Gamma^{th}$.

Algorithm1: Deep deterministic policy gradient (DDPG)

Initialization

Initialize: Critic network $Q(s_i, a_i)|\theta^Q$, Actor network $\mu(s|\theta^\mu)$, memory buffer D, weight parameters θ^Q and θ^μ, target network critic$\mu'(s|\theta^{\mu'})$, actor $Q'(s, a|\theta^{Q'})$ with $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.

Learning

for episode 1, E **do**

Initi: a random process $\mathbb{N}$ in the action exploration strategy

Imput initial SD-WAN and environment observation s_i

for $t = 1, T$ **do**

Select action $a_t = \mu(s_t|\theta^\mu) + \mathbb{N}_t$ based on current $\mu(s_t)$ policy and exploration noise.

Execute action a_t to get the reward r_t and new state s_{t+1}

Store transition (s_t, a_t, r_t, s_{t+1}) in D.

Sample random mini batch of $\mathbb{N}$ transition (s_i, a_i, r_i, s_{i+1})

Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.

Update online critic network by minimizing loss

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i)|\theta^Q)^2.$$

Update the actor policy using the sampled policy gradient

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}.$$

Update target network (Actor, Critic).

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\,\theta^{\mu'}$$
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\,\theta^{Q'}$$

end for

end for

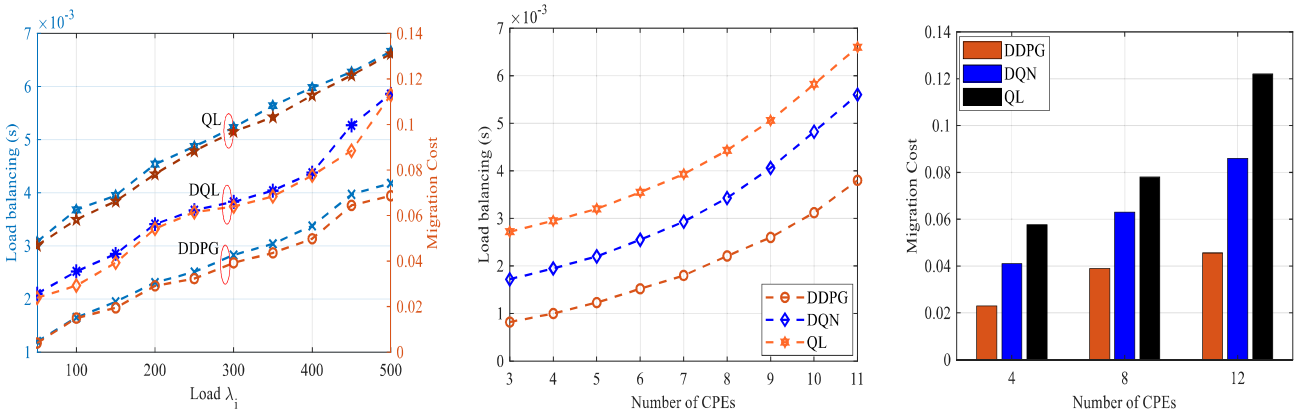
Reward Function: To guide the agent towards an optimal policy, reinforcement learning is based on the reward function. At each time step t, we reward SD-WAN controller (agent) based on our objective. However, our goal is to minimize both LB and cost. Thus, the reward function is designed as

$$r_i = \sum_{j=1}^{m}\sum_{j'>j}^{m} \left|\Gamma_{C_j} - \Gamma_{C_{j'}}\right| f_{LB} + \sum_{j=1}^{m} \Im_{C_j} f_\varphi \qquad (16)$$

where $f_{LB/\varphi} = 1$ if the flow has migrated and the LB factor has declined, $f_{LB/\varphi} = 0$ if the condition is not satisfied.

III. Simulation Results

To simplify our system, we consider that two controllers orchestrate the data plane, composed by a number of HQ such as 2, 6 and 10. In particular, communication between the site branch and the various HQs is established by means of overlapping tunnels. Similar to [9], we apply the Generic Routing Encapsulation (GRE) tunnelling protocol to conduct the overlay tunnels. We consider that the data plane is managed by a single controller at the beginning and migrates flows to another adjacent controller when it is congested. Moreover, we inject a considerable number of *Packet_In* messages on a specific controller for the unique reason of creating an overload at the control plane. Finally, we assume shift move operation [16] to migrate flow into an underloaded controller. The simulation was realized using Matlab (R2017a) running on PC Dell, 2,8Ghz @ Intel core i7-7600U, 16 GB. The main simulation parameters are provided in Table I. In fig. 4a, we show the impact of the *Packet-In* messages λ_i by CPEs on the load balancing and migration cost using different RL algorithms. A common observation in fig.4a is that when the number of *Packet-In* messages increases, the initial controller becomes saturated $L_1 > \omega_1$ and unable to manage the data plane. Therefore, SD-WAN controller selects the CPE with maximum traffic and migrate the flow to an appropriate underloaded controller $L_2 < \omega_2$. In addition, increasing λ_i from CPEs, increases the migration cost. This can be explained by the fact that the overloaded controller communicates and exchanges messages with another controller to receive the migration agreement, thus increasing the processing time.



(a) (b) (c)

Fig.4. Impact of number of flow sent by CPEs, numbers of CPEs on load balancing and migration cost, with different RL.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Flow (*Packet_In* messages) λ_i	[50-500]
Capacity Processing	1800
Response Time of controller Γ_{C_j}, ω_j	0.0065
Memory size D/Mini batch size	10^5/256
Actor/Critic learning rate	0.001
Reward discount	0.98
Weight factor and operation migration cost	0.6; 20$/s

As can be easily shown from fig.4b, the deployment of several CPEs in data plane leads to increase the load balancing in control plane. Obviously, increasing CPEs causes an increase in the number of *Packet-In* messages λ_i, consequently the controller is overloaded. We can also see from fig.4c, DDPG significantly outperforms LB problem compared to Q-learning (QL) and DQN. In fact, DDPG minimizes the load balancing perfectly because it quickly produces an effective migration policy compared to other alternatives. Also, QL and DQN tend to choose CPEs where the flow is inconsistent or missing. Fig.4c shows the performance comparison between DDPG, DQN and QL on the migration cost by varying the number of CPEs. Clearly, the number of CPEs migrated to the underloaded controller increases, which raises the cost of the migration operation. Nevertheless, The DDPG approach gives better results by minimizing migration costs. Technically, DDPG learns faster to choose the CPEs concerned by the migration.

IV. Conclusion

In this paper, we propose a novel approach based DDPG to optimize both load balancing and migration cost in SD-WAN solutions. Compared to traditional RL, our method guarantees quick convergence to an optimal solution by minimizing the load balancing. However, DDPG agent selects suitable CPEs to migrate flows into the underloaded controller. The reliability of CPEs to migrate flow is not considered in this work. In addition, the number of controllers can have an impact on the load balancing. In the future, we will investigate the load balancing in the virtual channel between CPEs, which is a major issue in SD-WAN, by using multi agent DDPG.

REFERENCES

[1] Du, C., Xiao, J., Guo, W.: Bandwidth constrained client selection and scheduling for federated learning over SD-WAN. IET Commun. Vol.6, pp. 187– 194, 2022.

[2] M. Tanha, D. Sajjadi, R. Ruby and J. Pan, "Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined wans," *in IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991-1005, Sept. 2018.

[3] K. S. Sahoo, P. Mishra, M. Tiwary, S. Ramasubbareddy, B. Balusamy and A. H. Gandomi, "Improving End-Users Utility in Software-Defined Wide Area Network Systems," *in IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 696-707, June 2020.

[4] A. Filali, Z. Mlika, S. Cherkaoui and A. Kobbane, "Preemptive SDN Load Balancing With Machine Learning for Delay Sensitive Applications," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 15947-15963, Dec. 2020.

[5] Z. Guo, S. Dou, Y. Wang, S. Liu, W. Feng and Y. Xu, "hybridflow: Achieving Load Balancing in Software-Defined wans With Scalable Routing," in *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5255-5268, Aug. 2021.

[6] K. Yang, D. Guo, B. Zhang and B. Zhao, "Multi-Controller Placement for Load Balancing in SDWAN," *in IEEE Access,* vol. 7, pp. 167278-167289, 2019.

[7] S. Troia, F. Sapienza, L. Varé and G. Maier, "On Deep Reinforcement Learning for Traffic Engineering in SD-WAN," *in IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2198-2212, July 2021.

[8] Z. Duliński, R. Stankiewicz, G. Rzym and P. Wydrych, "Dynamic Traffic Management for SD-WAN Inter-Cloud Communication," *in IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1335-1351, July 2020.

[9] K. S. Sahoo, P. Mishra, M. Tiwary, S. Ramasubbareddy, B. Balusamy and A. H. Gandomi, "Improving End-Users Utility in Software-Defined Wide Area Network Systems," *in IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 696-707, June 2020.

[10] J. Pei, P. Hong, M. Pan, J. Liu and J. Zhou, "Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-

Enabled Networks," *in IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263-278, Feb. 2020.

[11] Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi and Y. Mao, "Deep Reinforcement Learning for Load-Balancing Aware Network Control in iot Edge Systems," *in IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1491-1502, 1 June 2022.

[12] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE wireless communications*, vol. 24, no. 3, pp. 146–153, 2016.

[13] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang and K. Chen, "Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks," *in IEEE Transactions on Industrial Informatic*s, vol. 15, no. 3, pp. 1658-1667, March 2019.

[14] D. Xie and X. Zhong, "Semicentralized Deep Deterministic Policy Gradient in Cooperative starcraft Games," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 4, pp. 1584-1593, April 2022.

[15] J. Cui, Q. Lu, H. Zhong, M. Tian and L. Liu, "A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time," in IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1197-1206, Dec. 2018.

[16] Guo S, Zhang X, Zheng Y, Du Y. An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning. Sensors. 2020; 20(2):426.

ACKNOWLEDGMENT

This paper was financially supported by the Project "Network of excellence in applied research and innovation for doctoral and postdoctoral programs" / InoHubDoc, project co-funded by the European Social Fund financing agreement no. POCU/993/6/13/153437.