Load Balancing with Safe Reinforcement Learning

Lam Dinh, Pham Tran Anh Quang, Jérémie Leguay

Equilibrage de charge avec un apprentissage par renforcement sûr

Lam Dinh [1] et Pham Tran Anh Quang[1] et Jeremie Leguay[1]

[1] *Huawei Technologies Ltd., Paris Research Center, France*

Les algorithmes d'apprentissage par renforcement profond (DRL) ont récemment montré des progrès significatifs dans l'amélioration des performances des réseaux. Néanmoins, leur application reste limitée en l'absence d'exploration et de prise de décision sûres. Pour résoudre ce problème, nous proposons un algorithme d'équilibrage de charge sûr pour les réseaux définis par logiciel (SD-WAN), qui s'appuie sur de l'apprentissage par renforcement profond (DRL) combiné à une fonction de barrière de contrôle (CBF). Il projette heuristiquement les actions vers des actions sûres pendant l'entraînement et le test, et il guide l'apprentissage stochastique vers une politique sûre. Nous avons réussi à implémenter la solution sur GPU pour accélérer l'entraînement d'environ 110 fois et effectuer des mises à jour du modèle en quelques secondes, ce qui rend la solution déployable en pratique. Nous montrons que notre approche offre une qualité de service (QoS) quasi optimale en terme de latence de bout en bout, tout en respectant les exigences de la sûreté liées aux contraintes de capacité des liens.

Mots-clefs : Software Defined-Wide Area Network(SD-WAN), Deep Reinforcement Learning (DRL), Control Barrier Function (CBF).

1 Introduction

Many enterprises are adopting Software Defined-Wide Area Network (SD-WAN) technologies to trade-off between cost-effectiveness and Quality-of-Service (QoS) satisfaction. Relying on a network overlay, this architecture allows businesses to interconnect multiple sites (enterprise branches, headquarter, data centers) without the need to deploy their own physical infrastructure, making it cost effective. The key enabler of SD-WAN is based on the decoupling of the control plane and data plane which facilitates traffic engineering and queuing policies to meet Service Level Agreement (SLA) requirements. At a slow pace, the controller maintains policies, while access devices make real-time decisions for every flow.

Deep Reinforcement Learning (DRL) is a promising approach to optimize network utility under the umbrella of *experience-driven networking*. Since, several single-agent and multi-agent DRL solutions have been proposed to tune queues and load balancing policies to satisfy QoS requirements or minimize congestion [HZPa22] . However, most of the literature only focuses on off-policies and their performance once training has converged, without paying attention to safety during both learning and testing. Taking those issues into consideration, this work seeks to complement current DRL-based load balancing solutions with an additional safety shield. Based on the safe learning approach primarily presented in [ACE$^+$19], we propose a safe load balancing solution. The contributions of our work are the following. **Firstly**, we describe the target SD-WAN system and formulate a load balancing problem to minimize the average tunnel latency. **Secondly**, we design a dedicated Control Barrier Function (CBF) based on local search to deliver safety on top of gradient-based DRL algorithms (e.g., off/on policy learning). **Finally**,we compare our solution to traditional learning algorithms (e.g., DDPG, PPO) where safety is only handled in the reward function, without any strict guarantees. We show that the QoS obtained is very close to the optimal solution derived from a Non Linear Programming (NLP) model solved with the SCIP solver. In terms of execution time, we implemented DRL-CBF algorithms on GPU and managed to accelerate training by roughly 110x times and achieve model updates for on-policy methods within a few seconds, making the full solution practical.

2 System model and problem simulation

Figure 1 presents a typical SD-WAN use case where the headquarter and 3 branches of an enterprise are interconnected either via an Internet connection and a Multi-Protocol Label Switching (MPLS) private line. Traffic is issued by applications at both headquarter and branches. 6 OD (Origin-Destination) flows, also called *tunnels*, are considered, one per headquarter and branch pair in each direction. Each tunnel has two paths for Internet and MPLS. A Load Balancer (LB) agent at each Access Router (AR) splits the traffic according to the policy received by the centralized controller.

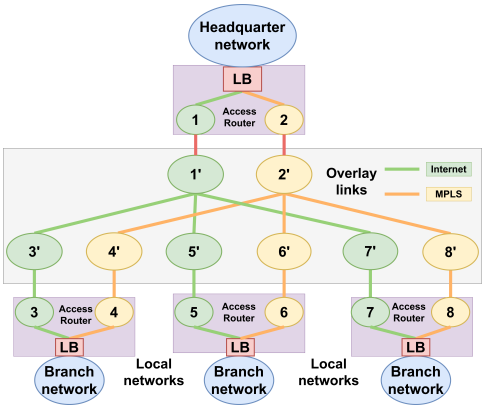


FIGURE 1 – SD-WAN network with an headquarter and 3 branches.

Let formally consider a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Each tunnel k in a set of tunnels K can use a set of *candidate paths* denoted as $\mathbb{P}_k$ (e.g., Internet and MPLS) to load balance traffic. Each edge e carries an instantaneous load l_e and has a capacity c_e. Let denote T^k the traffic demand of tunnel k at time t. Each LB agent applies at time t a split ratio x_p^k for each tunnel k over each path $p \in \mathbb{P}_k$ ($x_p^k \in [0,1]$ and $\sum_{p\in\mathbb{P}_k} x_p^k = 1 \; \forall k \in K$). The delay on each path p for a tunnel k is denoted d_p^k and the tunnel delay, denoted $d^k = \max_{p\in\mathbb{P}_k} d_p^k$.

Problem formulation. The main objective is to derive an optimal load balancing policy so that the SD-WAN overlay delivers the best QoS. Besides, in order to prevent high link delays, a common practice is to enforce a Maximum Link Utilization (MLU) $\mu \in [0,1]$ over all links. In this case, optimal load balancing policy is the solution for the following optimization problem :

$$\min_{x_p^k} \quad \frac{\sum_{k\in K} d^k}{K} \tag{$\mathscr{P}$}$$

$$\text{s.t.} \quad \sum_{k\in K} \sum_{i=0, p\in\mathbb{P}_k}^{|p|} T^k . x_i^k \leq \mu . c_e \quad \forall e \in E \tag{$\mathscr{C}_0$}$$

$$\sum_{i=0}^{|p|} x_i^k = 1 \qquad \forall x_i^k \in [0,1] \tag{$\mathscr{C}_1$}$$

$$d_e \geq \frac{1}{c_e - \sum_{k\in K}\sum_{p\in\mathbb{P}_k} l_p^k} \quad \forall e \in E \tag{$\mathscr{C}_2$}$$

where problem ($\mathscr{P}$) minimizes the average tunnel delay under several constraints. Constraint ($\mathscr{C}_0$) guarantees that the traffic over each edge e in the network is kept under the MLU. Constraint ($\mathscr{C}_1$) ensures that splits ratios sum to 1. To obtain an optimal solution as a benchmark (i.e., NLP solution), the following constraint ($\mathscr{C}_2$) is added to Problem $\mathscr{P}$ so that the link delay is computed according to M/M/1 queuing model. ($\mathscr{C}_2$) is crucial for the NLP solution to make ($\mathscr{P}$) solvable, but our safe learning-based algorithms do not need to know ($\mathscr{C}_2$) in advance. It highlights the advantages of our algorithm over NLP solution.

3 Learning-based and safe load balancing

3.1 Learning-based optimization

Our optimization problem can be formulated as a Markov Decision Process (MDP) which is defined by the tuple $\langle \mathscr{S}, \mathscr{A}, \mathscr{R}, \mathscr{T}, \gamma \rangle$ where $\mathscr{S}$ represents the set of states, $\mathscr{A}$ is the set of available actions, $\mathscr{R} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \rightarrow \mathbb{R}$ is the reward function which gives the reward for the transition from one state to another given an action, $\mathscr{T} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \rightarrow [0,1]$ is the transition matrix and $\gamma \in [0,1]$ is the discount factor. To solve the MDP associated with problem ($\mathscr{P}$), we consider the observation space or state s_t as the set of traffic demands T^k for all tunnels $k \in K$. The action space is determined as the set of split ratios x_p^k for all paths $p \in \mathbb{P}_k$ of each tunnel $k \in K$. Our reward function is designed to be the weighted sum of the average delay and the MLU (i.e., μ) as $r_t(s_t, a_t) = -\sigma \frac{\sum_{k \in K} d_{k,t}}{|K|} - (1-\sigma)\mu$, where $\sigma \in [0,1]$ emphasizes the importance of the average tunnel delay over a low MLU. While this reward cannot guarantee itself a hard safety, it guides RL agent in learning a policy which is both QoS optimal and safe after convergence. The general optimization procedure is detailed here [DQL24]. In order to enforce safe exploration during learning and safe policy execution during testing, we implemented a safety block, which is based on a CBF function [ACE+19], on top of DRL algorithms. The details of our algorithm are shown below.

3.2 Safe policy exploration and exploitation

The CBF function serves as a projector to convert the *proto-policy* π_θ, which is the parameterized actor network from which unsafe actions might cause congestion, into a *safe policy* π^{CBF}. Its main objective is to guarantee that the MLU μ remains below 1 (i.e., $\mu(a^{CBF}) \leq 1$, $\forall a^{CBF} = \pi^{CBF}(.|s)$). Following the safe policy π^{CBF}, each CBF action is determined according to the following optimization problem :

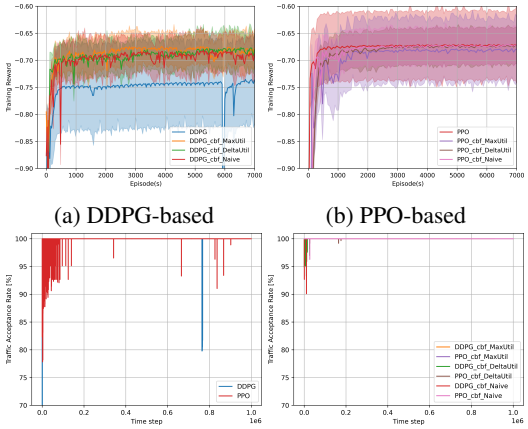
$$\begin{aligned} & a^{CBF} = \underset{a_t^{CBF}}{argmin} \left\| a_t^{CBF} - a_\theta \right\|_1 \\ s.t. \quad & a^{CBF} \in \mathscr{A} \\ & \mu(a^{CBF}) \leq 1 \end{aligned} \tag{1}$$

CBF function. In principle, given a proto-action, the CBF stochastically attempts to generates N safe actions within a neighbourhood of radius δ_s. The generation of actions is based on three different policies where the information with regards to the link utilization of each path p in the tunnel k is exploited : ***Naive policy*** randomly picks any tunnel $k \in K$. For each selected tunnel, a random value $\varepsilon \sim Uniform(0, \delta_s)$ is added/subtracted to current split ratios given by the proto-action on each path $p \in \mathbb{P}_k$. ***DeltaUtil policy*** selectively focuses on tunnels where the difference between their highest path utilization and lowest path utilization is greater than a certain threshold. In our case, a threshold on the difference of 50 % is chosen. ***MaxUtil policy*** inherits the principles of the ***DeltaUtil policy***, but it uses a different criteria for selecting tunnels. Specifically, any tunnel k that has a path load utilization above a threshold of 100 % (e.g., $\exists p \in \mathbb{P}_k \mid \mu_p \geq 1$, which is unsafe) will be the target for proto-action modification. After generating a large number of actions around a proto action, a feasible action (i.e., MLU is below 100%) is selected in such a way that its distance to the original proto action is the smallest. The returned action is therefore heuristically safe and helps learning better policies.

4 Results and discussions

We implemented the solution on a server composed by a CPU Intel® Xeon® Platinum 8164 and a GPU NVIDIA® Tesla V100. As local search algorithms can be massively parallelized, we implemented them with CUDA libraries so that they fully benefit from all GPU cores available.

Training performance. Figures 2a and 2b compares the average training reward during each episode for DDPG, PPO, DDPG-CBF and PPO-CBF, respectively. It highlights that PPO typically achieves better rewards than DDPG. Besides, the DDPG agent tends to be trapped into local optimum policies, which can lead to sub-optimal performance. When safe training using CBF is applied, they demonstrate that both DDPG-CBF and PPO-CBF algorithms attain decent training reward and successfully converge. In terms of



(a) DDPG-based (b) PPO-based

(c) Acceptance (no CBF) (d) Acceptance (CBF)

FIGURE 2 – Training performance.

Algorithms	Mean/std Delay	Mean/std MLU	Ave. calcul time per training step (CPU)	Ave. calcul time per training step (GPU)
NLP	0.63 ± 0.18	0.81 ± 0.08		
DDPG	0.7 ± 0.24	0.84 ± 0.09	0.01 (s)	0.01(s)
PPO	0.66 ± 0.17	0.82 ± 0.08	0.02 (s)	0.02 (s)
DDPG-Naive	0.74 ± 0.3	0.87 ± 0.11	11.45 (s)	0.15 (s)
DDPG-DeltaUtil	0.74 ± 0.22	0.85 ± 0.10	11.40 (s)	0.14 (s)
DDPG-MaxUtil	0.74 ± 0.37	0.84 ± 0.10	11.30 (s)	0.10 (s)
PPO-Naive	0.67 ± 0.17	0.83 ± 0.09	11.40 (s)	0.20 (s)
PPO-DeltaUtil	0.67 ± 0.17	0.83 ± 0.09	11.30 (s)	0.16 (s)
PPO-MaxUtil	0.68 ± 0.17	0.82 ± 0.09	11.20 (s)	0.14 (s)

TABLE 1 – Testing performance.

traffic acceptance rate which is related to safety requirements, Figure 2c illustrates the percentage of the total traffic that is accepted due to the link capacity constraints. Both algorithms encourage policy exploration at early stages in training, which accidentally causes severe link congestion and traffic rejection. Although DDPG-CBF achieves a better reward than its non-safe version, its training curve is slightly unstable compared to the training curves of PPO-CBF. More importantly, Figure 2d, which shows the training performance for the three CBF functions, illustrates that traffic acceptance rate during learning is significantly improved compared to the case without CBF, as illustrated in Figure 2c.

Testing performance. In order to show testing performance, we generated 100 random traffic samples from each tunnel and used various learning algorithms and an optimal solution using NLP to benchmark the average delay and the MLU. The results are then displayed in Table 1. It reveals that near-optimal delay performance are obtained using conventional DDPG and PPO learning algorithms when compared to optimal results of NLP. Besides, the MLU during testing is safely kept below 1, resulting in no traffic rejection. When the safety CBF layers are applied on top of current off/on policy learning algorithms, both DDPG-CBF and PPO-CBF also reach near-optimal delay performance and PPO-CBF performs much better than DDPG-CBF. Furthermore, safety constraint in the testing phase is always respected (MLU belows 1). Table 1 also shows the benefits of our GPU for training acceleration. It can be observed that a feasible action is found in around $0.1(s)$ when using GPU, compared to more than $11(s)$ using CPU.

5 Conclusion

We presented a novel approach combining the DRL and a CBF to guarantee safe load balancing in the context of SD-WAN. We show that on-policy optimization based on PPO achieves better performance than off-policy learning with DDPG. We implemented all the algorithms on GPU to accelerate training by approximately 110x times and achieve model updates for on-policy methods within a few seconds, making the full solution practical. Future works along these lines include the integration with a network simulator and a testbed for a more realistic performance evaluation.

Références

[ACE+19] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control Barrier Functions : Theory and Applications, March 2019.

[DQL24] Lam Dinh, Pham Tran Anh Quang, and Jérémie Leguay. Towards safe load balancing based on control barrier functions and deep reinforcement learning. (arXiv :2401.05525), January 2024.

[HZPa22] Omar Houidi, Djamal Zeghlache, Victor Perrier, and et al. Constrained Deep Reinforcement Learning For Smart Load Balancing. In *IEEE CCNC*, pages 207–215, 2022.