Scalable Reinforcement Learning for Dynamic Overlay Selection in SD-WANs

Alessio Botta, Roberto Canonico, Annalisa Navarro, Giovanni Stanco, Giorgio Ventre
DIETI Department, *University of Napoli Federico II*, Naples, Italy
{*alessio.botta, roberto.canonico, annalisa.navarro, giovanni.stanco, giorgio.ventre*}@*unina.it*

***Abstract*—SD-WAN promises distributed enterprises to satisfy their dynamic communication requirements over the public Internet with a substantial cost reduction and enhanced performance compared to dedicated lines. It builds interconnections between users or applications in remote sites by exploiting all available transport connections (e.g. Internet, MPLS, ...), but how to combine them to enhance communication performance is still an open challenge. Previous work investigated the use of Reinforcement Learning in the SD-WAN control logic to solve this problem, but they only considered simple scenarios consisting of two sites connected by two paths. In this paper we move a step forward and pose the question of whether such a promising approach can scale to WANs spanning multiple distributed sites connected through several paths. We first conduct an analytical study of the complexity of Reinforcement Learning that considers the increase of action and state spaces when the number of sites and paths grows. We then propose a solution based on Multi-Agent Reinforcement Learning (MARL) that helps reducing the overall complexity by leveraging an agent for each site. Finally, we show the effectiveness of our solution with real experiments in an emulated environment, showing that not only it is viable, but it also achieves a reduction in network policy violations, latency, and transit costs in a multi-site scenario.**

***Index Terms*—SDN, SD-WAN, Traffic Engineering, Reinforcement Learning, Scalability**

I. Introduction

Traditional networks are inherently static and lack fast adaptation mechanisms. Although change can be eventually achieved, current Traffic Engineering (TE) solutions take time to be implemented and are unsuitable to be deployed over wide geographic regions [1]. Software Defined Wide Area Network (SD-WAN) is a new emerging technology that can be leveraged to enable the adaptation of networks on both temporal and geographic scales. SD-WAN is in fact designed to guarantee secure and reliable communications between different sites distributed on a large geographical region and connected through multiple up-links [2]. SD-WAN guarantees fast network reconfiguration to meet application QoS requirements, taking advantage of real-time monitoring and centralized control.

Despite the unquestionable benefits of Software Defined Networking (SDN) in solving classical TE problems [3], many challenges have to be tackled when it comes to WANs. One is for example guaranteeing scalability and robustness. Traditional, fully centralized controller with global knowledge and managing the entire network is complex and also represents a single point of failure. For this reason, distributed controller architectures have been proposed, where each controller has only a local view and manages a subset of the overall architecture, improving scalability and reducing the impact of failures [4]. The challenges also extend to the data plane. Traditional enterprise networks use a hub-and-spoke topology. In this architecture, remote sites are not directly interconnected, but rely instead on the central site as a conduit for all traffic, leading to bottlenecks and single point of failure problems [5]. This issue is further exacerbated by the increasing adoption of cloud-based services, as traffic from remote sites must traverse the central site before being directed to the cloud, resulting in a significantly increased latency.

Another important challenge is how to realize network automation in practice: in the management plane, network administrators exert general high-level network intents (also referred to as network policies) that specify the desired behavior for a set of traffic categories or applications, taking into account factors such as performance, Quality of Service (QoS) or transit costs. The problem of inferring the appropriate network configuration to effectively meet these intents remains an open area of research. For instance, if a policy stipulating that latency-sensitive traffic must not experience delays exceeding 300 milliseconds is established, any deviation from this threshold should trigger immediate steering of the traffic to an alternate WAN transport that is able to guarantee optimal network performance.

In other words, an SD-WAN should be *self-adaptive*, modifying its behavior in near real-time and in a dynamic fashion. This adaptation is in response to feedback, such as poor performance, and to the desired optimization goal, such as minimizing policy violations. The application of Reinforcement Learning (RL) techniques to address this requirement has been straightforward and proved to be very promising in this scenario. RL is in fact a field of machine learning for training agents to make decisions by maximizing a reward signal. The agent interacts with the environment, taking actions and receiving rewards or penalties, and learns to adjust its behavior accordingly. In dynamic environments such as networks, where the conditions or rules may change over time, RL allows the agent to respond quickly, by continuously learning from new interactions and experiences. One of the fundamental limitations of RL that hinders its practical application is scalability. This arises when the state and action spaces become very large or when the environment is highly complex. As the number of states and actions increases, the amount of data and computation required to learn an optimal

ISBN 978-3-903176-57-7 ©2023 IFIP

policy becomes infeasible [6]. Although there are cases where SD-WAN has been used to interconnect more than 1000 sites spread around the world [7], current solutions based on RL have only been tested in simple network environments.

In this paper, we study the feasibility of RL for controlling an SD-WAN in more complex scenarios with multiple sites connected through multiple overlays. We show that the number of states and actions explodes as soon as the network architecture becomes more complex if using a single agent for the entire network. To solve this problem we propose Multi Agent Reinforcement Learning (MARL), which has been recently used to solve various issues in new emerging network scenarios [8]. We show that MARL helps reduce complexity through decentralized control: the state and action spaces can be reduced distributing the decision-making process among multiple agents, making it easier for the individual agents to learn the correct behavior. We also show how it can be used in a fully distributed architecture of controllers, with a controller for each site deciding the overlay to use to reach any other site of the enterprise WAN. We also test our solution in an emulated scenario with two sites and three overlay connections (or simply paths) among them. A RL agent connected to the first site is responsible for deciding the overlay to use to route traffic to the second site. Results show that the RL agent helps reduce network policy violations, guaranteeing low end-to-end latency, and reducing monetary costs.

The paper is organized as follows. Sec. II describes the state of the art about TE in SD-WAN with particular attention to solutions applying RL. In Sec. III we explain in detail the SD-WAN scenario under consideration, while in Sec. IV we describe the proposed solution based on RL. Sec. V addresses the feasibility of the proposed solution when the number of overlays and edges grows. Sec. VI presents experiments and results while Sec. VII draws conclusions and discusses future works.

II. Related Work

In this section, we discuss work related to TE in SD-WAN, with particular attention to RL-based solutions.

A TE SD-WAN application has been proposed, consisting of a monitoring system that raises an alert anytime performance degradation is detected on the current overlay, and another module responsible for switching overlay anytime the alert is received [9]. The application is evaluated in two testbeds - one with two sites in a metropolitan city and one with four sites in an emulated environment - showing an improvement in service availability and delay for time-sensitive traffic. A semi-distributed, intent-based system for optimizing routing policy in an SD-WAN scenario comprising a central headquarters site and three branch sites interconnected in a hub-and-spoke topology has also been presented [10]. The proposed model selects the appropriate overlays for applications based on their specific requirements or network intents, such as minimizing latency, congestion, or costs. The model utilizes a local minimum search algorithm in conjunction with a traffic prediction module to determine the optimal set of overlays for routing traffic.

A novel implementation of Multi-Path Transmission Control Protocol (MPTCP) specifically tailored for SD-WAN environments, referred to as WAN-aware MPTCP, has recently been presented [11]. This implementation aims to optimize the utilization of multiple, heterogeneous WAN transports through aggregation, while also providing fast failure recovery for applications. The authors show the effectiveness of their approach in an emulated testbed comprising 2 branch sites and 5 tunnels, as well as a real-world testbed consisting of 3 branch sites and 2 tunnels. Authors of [12] propose the utilization of RL techniques in SD-WAN for the efficient implementation of network policies. They conduct experiments utilizing various Deep Reinforcement Learning (DRL) algorithms on a dataset of end-to-end delay time series, collected from an SD-WAN scenario comprising two interconnected sites with multiple overlays. Results show that the Deep Q-Learning algorithm exhibits the fastest convergence time and provides evidence for the effectiveness of RL in achieving policy constraints.

In [13], the authors employ DRL to determine the flow-splitting ratios for balancing traffic over multiple paths with the aim of minimizing the latency of tunnels while satisfying capacity constraints. This approach is evaluated in an SD-WAN scenario consisting of a central headquarters site and three branch sites that are interconnected through both Multi-Protocol Label Switching (MPLS) and the Internet. The proposed approach successfully achieves low end-to-end delay and reduces link capacity violations. The authors evaluate the model by emulating the environment using queuing theory, specifically by estimating the delay based on link capacities and allocated bandwidth. In [14], the authors present a Multi-Agent Reinforcement Learning (MARL) based approach for scheduling flows in an SD-WAN scenario based on their traffic category. The objective is to meet the demands in terms of throughput or delay for each flow. The evaluation of the proposed approach is conducted within a hub-and-spoke topology, comprising a central headquarters site and five branch sites interconnected. The results demonstrate that by using a hybrid approach, the cooperating agents are capable of effectively accommodating traffic demands and perform better with respect to the fully distributed approach.

In [15] we proposed a dynamic routing framework for SD-WAN based on RL with the ability to steer traffic on one of the two alternative overlays connecting two branch sites of a company based on network measurements. The objective of this framework is to meet performance and transit cost goals, by adapting to highly variable network conditions. We conducted experiments in an emulated environment, observing a reduction in the number of policy violations compared to a benchmark approach. In this paper, we extend our previous work considering a more complex scenario with multiple sites and overlays.

Previous studies have demonstrated that RL is a highly promising technique for improving the behavior of SD-WAN as a means of realizing network policies, balancing traffic

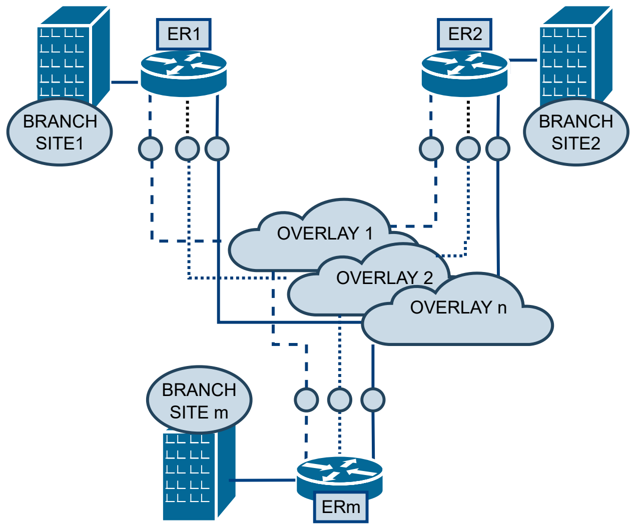


Fig. 1: A typical WAN: m sites connected through n overlays.

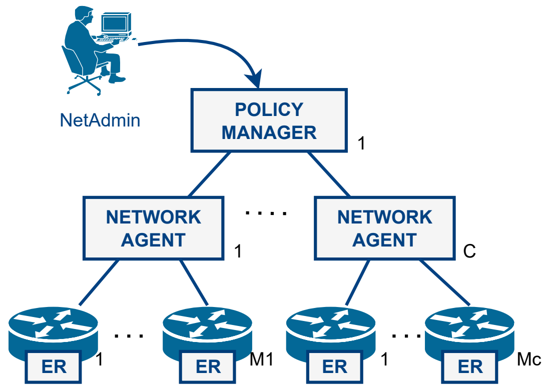


Fig. 2: SD-WAN reference architecture: each network agent configures one or more ERs to reach the asserted network policy.

flows, reducing service delays, and so on. However, limited works have investigated the scalability of such approach, evaluating it only in simplistic scenarios such as networks with a small number of sites, hub-and-spoke topologies, and theoretical models. In this study, we argue that scalability is a critical issue that must be addressed to enable the application of RL in real-world scenarios. We consider a full mesh topology and we propose MARL as a potential solution. To the best of our knowledge, there is a single work that has introduced MARL in an SD-WAN scenario [14], but it only uses it for traffic flow scheduling. In contrast, we propose to use it to solve the dynamic overlay selection problem.

III. Problem statement

Fig.1 shows a typical WAN: m branch sites (e.g. enterprise sites) are connected through n possible up-links (also called overlays) by means of Edge Routers (ERs). ERs collect traffic flows from branch sites and route them to the destination through the available overlays. Overlays can be of different types (e.g. MPLS, Internet, ...) and can also be under the administration of different Internet Service Providers (ISPs). For this reason, in most SD-WAN use cases, enterprises are not able to control the overlays and their intra-domain routing. The only possible control action is executed on ERs [16] [17]. ERs are in fact part of the enterprise network and can be directly instructed by a controller to perform some actions. Examples of such actions include the utilization of per-flow or per-packet load balancing techniques through the combination of multiple overlay networks in order to aggregate bandwidth, duplication of packets for the purpose of ensuring their successful delivery or for error correction, and dynamic switching of paths for flows that are experiencing a sub-optimal performance [18].

In this paper, we address the challenge of dynamically responding to performance degradation in overlays by switching the one used for routing traffic when the current performance obtained is inadequate. To implement this near real-time control, we require feedback from the network to determine its current conditions. ISPs typically do not disclose information about their internal topology and the state of the links. Therefore, we employ a monitoring module that continuously assesses network conditions on the available overlays through a probing mechanism [19]. Other approaches have been presented in literature for this aim [20] which will be considered in future work. We also leverage a configuration module that translates a routing decision into its effective realization by instructing ERs to forward the selected flow through a specific overlay. This work will not address the monitoring module and its performance, as well as the configuration modules, since we covered it in detail in [15]. From now on, we assume that near real-time one-way delay for each overlay is known and that the routing decisions are directly translated into the appropriate network configuration.

SD-WAN solutions allow enterprise network administrators to define policies to meet network or application intents. Real-time network measurements are compared to the application or network requirements to select overlays eligible to route traffic meeting those constraints. In this paper, we consider two policies: the former - *policy A* - aims at achieving low latency while the latter - *policy B* - also targets commercial cost minimization (each overlay may have a different cost per traffic volume [21]). Both policies are threshold-based: a network administrator can set the threshold value to the maximum acceptable latency. The objective is to minimize the number of threshold violations (or alternatively maximize service uptime). In Fig. 2 we show the reference architecture. The network administrator interacts with a single management system to assert the traffic requirements; there may be one or more network agents that take routing decisions for one or more ERs based on those requirements.

IV. Reinforcement Learning solution

In this section, we present our solution based on Reinforcement Learning (RL). It relies on two actors: the agent and the environment. The agent interacts with the environment through actions and consequently changes its state and gets rewards or penalties based on the goodness of its behavior [6]. Thanks to continuous trial and error it learns about the environment and the best strategy to adopt in order to achieve a set of objectives without any prior knowledge about the environment.

Environment. The environment is the entire WAN, including sites, overlay links among sites, and access routers. In reality, there is no full visibility on the entire network: in fact, we consider the typical SD-WAN scenario in which the overlay networks are not under the control of the enterprise operating the SD-WAN. Hence, we have no knowledge about what is inside the "clouds" depicted in Fig. 1 (internal topology, link capacity, traffic traversing links, etc.). We only assume that we can monitor some network parameters - in particular, the end-to-end delay - from the edges of the overlays through probing mechanisms.

Rewards. While in typical RL contexts of application - such as gaming - rewards are directly returned by the environment, proper reward functions have been designed in this scenario. Basically, the reward is positive when the network policy in place is respected; instead, the reward is negative when the policy is violated. More in detail, the reward is a function of real-time monitored delay on the selected overlay in relation to the pre-set delay threshold.

Actions. For the same reason as before, TE policies can not be applied inside the overlay connections (e.g. manipulating routing decisions by changing link weights). The only control action can be executed by instructing ERs to select one of the available overlays to forward traffic for each destination site.

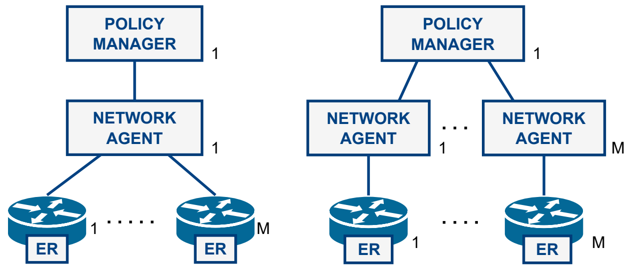
Among the different RL model-free methods, the ones that have proven to be most effective in WAN path selection are those based on the state-action-value function, in particular, the Q-learning [12].

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

As reported in (1), this algorithm is based on updating the Q-value $Q(s_t, a_t)$ - that estimates how good performing action a_t in a given state s_t is - according to the Bellman equation, where α is the learning rate that weights the current experiences versus the old ones, γ is the discount factor that weights the future rewards with respect to the current one, and r_t is the immediate reward received by the environment. The action with the highest Q-value at state s_t is chosen with probability ϵ while a random action is chosen with probability $(1-\epsilon)$. Parameters α and γ are tuned via grid search, while ϵ is at its maximum at the beginning so that the agent can learn fast, and then it decays at an exponential rate to a minimum value. The objective of Q-learning is to maximize the cumulative reward $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$.

V. Scalability evaluation

In this section, we model the RL solution firstly solving the problem of 2 sites and n alternative overlay connections between them, posing particular attention to the complexity



(a) SARL (b) MARL

Fig. 3: From a single Network Agent configuring all the ERs to multiple Network Agents that configure a single ER.

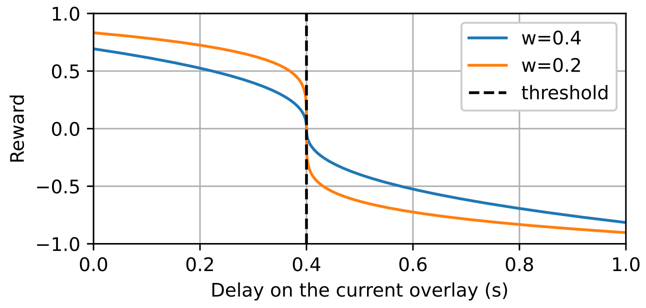


Fig. 4: Reward computation as a function of the experienced delay and the policy threshold.

of the solution as the number of overlays increases. We then consider the problem of m branch sites and n overlays. In this case, there are several possible solutions ranging from maximum degree of centralization, where a single agent makes decisions for all edge sites, to complete decentralization where there is a single agent for each edge site.

We study the two limit cases. One in which a single RL agent has a global view of the environment and configures all the ERs, i.e. Single Agent Reinforcement Learning (SARL) (Fig. 3a). The other one is based on MARL: there are multiple agents and each of them has only a local view and is responsible for configuring a single ER (Fig. 3b). We assume that the routing does not impact significantly the overlay performance, hence the choices made by the agents can not influence each other. Therefore a fully distributed MARL approach is proposed, in which the agents are totally independent and work concurrently. We evaluate the scalability of both approaches by increasing the number of overlays and sites.

A. Model for 2 sites and n overlays

The Q-learning scheme is defined by actions, states and rewards. Given n available overlays among branch sites, actions are defined as $[O(t)]$, where $O(t) = i$ if overlay i is chosen at time t and hence can assume n different values. Given $d_i(t)$, the delay measured on path i at time t, and T, the delay threshold set with the network policy, states are defined by the tuple $[O(t-1),\ x_i(t)\ ]\ \forall i \in [1, n]$, where $O(t-1)$ is the overlay used to route traffic at time t and hence is equal to the action at time $t-1$ and

$$x_i(t) = \begin{cases} 0, & \text{if } d_i(t) \leq T \\ 1, & \text{otherwise} \end{cases}$$

Dynamic reward functions are designed. The reward r at time t is calculated as follows:

$$r(t) = -\frac{\left|d_{O(t)} - T\right|^w}{d_{MAX}(t)} \cdot sign(d_{O(t)} - T) \qquad (2)$$

Where $d_{max}(t)$ represents the maximum delay experienced at time t and it is used for normalization, and w is an exponent with real values in range [0, 1] that controls the sensitivity of the reward to the deviations from the threshold. A high value of w will result in a low reward for small deviation from the threshold. On the contrary, low values of w emphasize small deviations. The value obtained is multiplied by the opposite of the sign of $(d_O(t) - T)$, resulting in a positive reward for delay below the threshold and a negative reward for delay above the threshold. The final rewards value is in the range $[-1\ ,1]$. For better understanding, Fig. 4 shows the possible reward obtained for different delay values fixing the threshold to $400ms$ and assuming a maximum experienced delay of $1s$. A delay below the threshold leads to positive rewards while a delay exceeding the threshold leads to negative rewards. The computation of the result for policy B is slightly different because it also considers the transit cost of the overlays. Hence, while the magnitude is the same as the one for policy A, the sign changes if multiple overlays are experiencing a delay under the threshold. In this case, if the overlay with the higher cost is used, the reward is negative. More formally, if $x_o(t) = 0$ but also $x_i(t) = 0$ $(i \neq O(t))$ and $cost_i < cost_{o(t)}$ reward is negative.

The reward computation has also to take into account the cost paid for the reconfiguration. Whenever there is a change in the overlay used, the service is impacted for a certain time. To prevent the agent from switching overlay continuously, a small negative penalty p is added to the reward function whenever the overlay is switched. The algorithm proceeds at a certain frequency f: every $\frac{1}{f}s$, the agent performs an action and receives a reward. Notice that the agent receives a reward even if $O(t) = O(t-1)$ (it did not switch overlay) to take into consideration possible changes in the environment state not dependent on the agent action. The frequency f is an important parameter that should be as small as possible but is limited by the time it takes for the system to detect a policy violation and effectively operate a reconfiguration.

We now evaluate the scalability of the RL solution increasing the number of overlays, taking into account the number of states, actions, and the consequent Q-table dimension. From now on, we will omit t in the mathematical notation.

1) 2 overlays: 2 are the possible actions, corresponding to choosing one of the 2 available overlays. States instead vary based on O, which represents the overlay currently in use and hence can have 2 possible values and x_i that for each of the 2 overlays can assume 2 possible values ($d_i \geq T$ or $d_i < T$). Hence, the number of states is $2(2^2)$. Since for the last combination - when both paths exceed the threshold - one of them may exceed it in a less severe way, there are two more combinations to be considered and the number of states becomes $2(2^2 + 2 - 1) = 10$. For easier understanding, we show the states in Tab. I. The Q-Table is 10 x 2 since there are 10 states and 2 actions (the Q-value is specified for each $state - action$ couple).

STATE	$d_1(t)$	$d_2(t)$	$O(t)$
s_1	$< T$	$< T$	0
s_2	$< T$	$< T$	1
s_3	$< T$	$> T$	0
s_4	$< T$	$> T$	1
s_5	$> T$	$< T$	0
s_6	$> T$	$< T$	1
s_7	$\gg T$	$> T$	0
s_8	$\gg T$	$> T$	1
s_9	$> T$	$\gg T$	0
s_{10}	$> T$	$\gg T$	1

TABLE I: States for 2 available overlays and 2 sites.

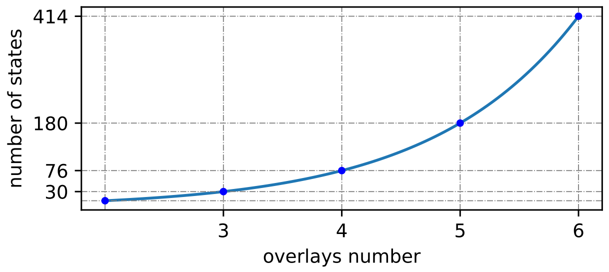


Fig. 5: Explosion of state space increasing the number of overlays.

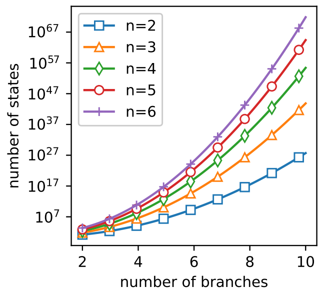
2) n overlays: generalizing, if there are n different paths between two ERs there will be $n(2^n + n - 1)$ possible states and n actions. Hence, Q-table dimension becomes $n^2(2^n + n - 1)$ x n. As can be observed in Fig. 5, the number of states increases exponentially with the number of available overlay links, but this approach is still feasible considering the maximum number of possible overlays to be 6 (it is unrealistic that a company site has more than 3 or 4 up-links).

B. Model for m sites and n overlays

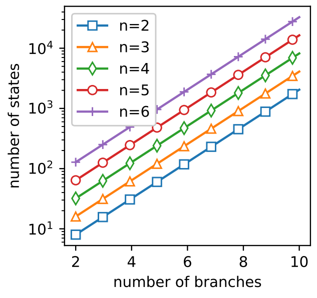
We will now consider a more general scenario with m branch sites that are connected in a full mesh. Actions are defined as the tuple $[o_{u,v,c}]$ $\forall u, v \in M : u \neq v$ and $c \in C$ where the general $o_{u,v,c}$ is equal to the overlay chosen to forward traffic flows of category c from site u to site v and hence has n possible values. We define d_i as the delay experienced on overlay i and T_c the delay threshold for traffic belonging to category c. Then we define:

$$x_{i,c} = \begin{cases} 0, & \text{if } d_i \leq T_c \\ 1, & \text{otherwise} \end{cases}$$

Finally, the state is defined as the tuple: $[o_{u,v,c}, x_{i,c}]$ $\forall u, v \in M : u \neq v$, $\forall i \in N$ and $\forall c \in C$, because with a centralized



(a) SARL



(b) MARL

Fig. 6: Managing complexity by transitioning from a single-agent to a multi-agent approach: reduction in the number of states.

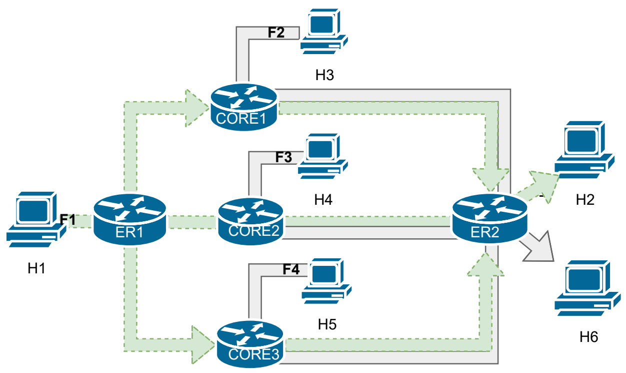


Fig. 7: Experimental setup: F1 (user traffic flow); F2, F3, F4 (background traffic flows).

approach a different overlay can be chosen for traffic between two sites with a specific traffic category, and also each traffic category has a different delay requirement.

1) SARL: With SARL, an action specifies an overlay for each category c of traffic and for each of the $m(m-1)$ couples of sites. With n possible overlays there are $n^{cm(m-1)}$ possible actions. States vary depending on the chosen path for each traffic category between a couple of sites and the delay with respect to the threshold set for each possible category, hence there are $n^{cm(m-1)}$ x 2^{nc} possible states.

2) MARL: With MARL, the model is the same as before but with u fixed, in fact, each agent only decides the overlay used for routing traffic from the branch it is responsible for towards each of the destination sites. Hence, each agent only decides among $n^{c(m-1)}$ actions. There are c possible categories; delay for each of the n overlays is compared to c thresholds; number of states is $n^{c(m-1)}$ x 2^{nc}. Fig. 6b shows the cardinality of the state space increasing the number of branches for different numbers of overlays, considering $c = 1$. Complexity soon becomes too high (Fig. 6a) with SARL, while MARL keeps complexity manageable, by exploiting multiple agents that take independent decisions, as shown in Fig. 6b. As the SARL approach has showcased complexity, moving forward we will solely focus on MARL in our experiments.

Flow	Maximum Load	Type	On-time	Off-time
F1 (UDP/TCP)	0.8 $Mbps$	constant	none	none
F2 (UDP)	2.6 to 3.7 $Mbps$	bursts	8 s	8 s
F3 (UDP)	2.6 to 3.7 $Mbps$	bursts	8 s	8 s
F4 (UDP)	2.6 to 3.7 $Mbps$	bursts	16 s	16 s

TABLE II: Parameters of traffic generation

Parameter	Value
Discount Factor (γ)	0.2

Exploration Rate (ϵ)	min = 0.001
	max= 0.8
	decay rate = 0.02

Learning Rate (α)	0.3
Weight (w)	0.2
Frequency (f)	0.5 s
Penalty (p)	0.2
Constant (k)	0.2

TABLE III: Parameters of MARL / Naive Algorithms.

VI. Experimental evaluation

MARL has proved to be a suitable approach to keep complexity manageable. In this section, we show its effectiveness in configuring an ER to meet the policy requirements.

A. Experimental setup

1) Topology: A simplified SD-WAN scenario is implemented using the Mininet emulator, as reported in Fig. 7. Two remote sites are connected through three different overlays. The former comprises H1 and ER1, while the latter comprises H2, H6, and ER2. The three overlays are modeled through links with specific nominal latency and capacity values and three switches to emulate propagation and queueing delay caused by the WAN transports. The capacity of the external links - i.e. the ones corresponding to the WAN connections - is set to 6 $Mbps$ to avoid bottlenecks in the edge sites. Conversely, the capacity of the internal links - i.e. the ones connected to the end devices - is set to 3 $Mbps$ and their latency is set to 25 ms. The ERs are emulated by OpenvSwitches that are connected to an SDN controller (Ryu controller) that acts as an intermediary layer between the switch and the agent, configuring the OpenvSwitch to choose an output interface that represents the egress for a specific overlay. On the contrary, it is assumed that the core switches are not controllable and hence are non Openflow switches.

2) Traffic generation: A traffic flow F1 (user traffic flow) is generated by H1 and directed to H2, simulating a communication between the two sites. The objective is to route this flow through the overlay that guarantees acceptable delay values, as specified by a pre-determined policy that sets the maximum acceptable delay. In addition to F1, other background traffic flows, F2, F3, and F4 are generated by H3, H4, and H5 to traverse the 3 alternative overlays. The characteristics of the traffic flows are reported in Tab. II. These flows represent traffic generated by other sites, or by any other not controllable source to emulate public WAN links. Background flows cause significant delay variation on the respective overlays in different time intervals. Traffic is in fact generated with bursts of

Link Load	Average Link Load	Maximum Link Load
Low	0.53	0.98
Medium	0.60	1.12
High	0.67	1.25
Very High	0.70	1.32

TABLE IV: Traffic load classes.

different duration and bitrate. The experiments were repeated increasing the bitrate of the background traffic bursts, causing more and more severe congestion during the on-times. The increasing bitrates caused the increase in the Average Link Load (AL). Experiments were conducted for AL values in the range [0, 1]. In the remainder, only AL values within the range [0.5, 0.7] are reported as they are considered relevant. Values smaller than 0.5 did not cause significant latency. On the contrary, values greater than 0.7 resulted in high levels of packet loss and were no longer relevant in terms of latency. In fact, these values corresponded to a Maximum Link Load (ML) greater than 1 during on-times, causing complete congestion of the bottleneck link. The background flows are based on UDP. A first round of experiments was conducted in which the useful traffic (F1) is also based on UDP. A subsequent series of experiments were also conducted using Cubic TCP for F1.

3) *Evaluation:* Results are evaluated by taking into account the latency experienced by F1, as well as its service uptime, which is defined as the percentage of time in which F1 is compliant with the policy (i.e. experiencing a delay below the set threshold). The results obtained by the RL approach are evaluated by comparing them to the ones achieved by a Naive Algorithm (NA) that at each iteration picks the path with the lowest latency by less than a constant k. This constant is set to prevent the algorithm from continuously switching overlay to avoid the "channel flipping problem". The parameters chosen for both algorithms are in Tab. III. The results are reported in terms of Average and Maximum Link Loads, conventionally referred to as Low, Medium, High, and Very High Link Load, as indicated in Tab. IV.

B. Policy A results

1) *UDP:* The latency experienced under various load conditions is presented for both the NA and the MARL approaches in Fig 8. It can be observed that, as the load increases, the average latency rises for both algorithms but MARL manages to keep it at smaller values. The interquartile range for the MARL approach is very low for light and medium loads, only increasing for high and very high LU, i.e. in overload conditions. Fig 9 shows the service uptime percentage. It can be observed that service uptime decreases as traffic load increases for both algorithms, but MARL manages to maintain a higher service uptime, except for very high load, in which packet loss is also observed.

To explain the behavior of the MARL agent, Fig. 10a illustrates the instantaneous latency for a short time interval in a single experiment with a medium load. The marked lines depict the instant latency experienced by alternative paths and

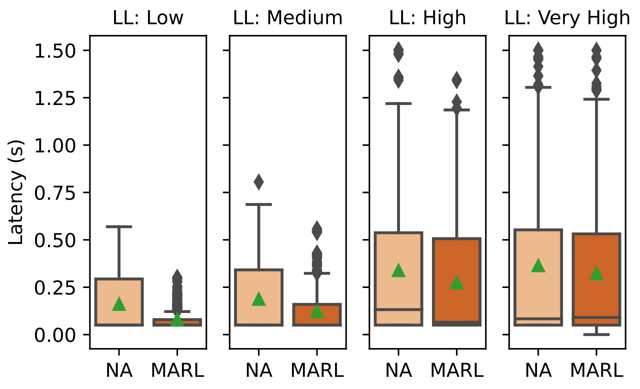


Fig. 8: Policy A - Latency achieved by MARL and NA for increasing Link Loads (UDP).

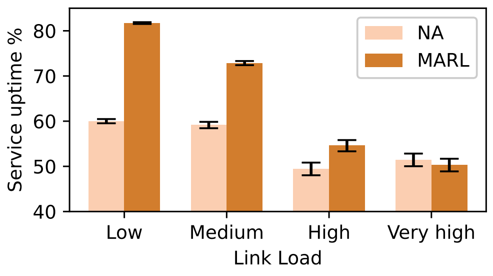
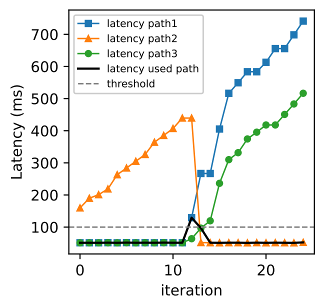


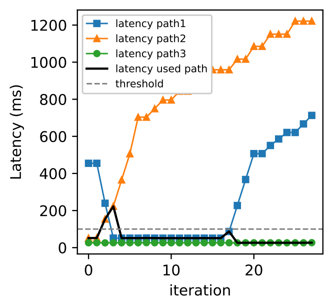
Fig. 9: Policy A - Service uptime achieved by MARL for increasing Link Loads (UDP).

the thicker line represents the actual latency experienced by the path chosen by the MARL agent, which overlaps with the line that corresponds to the used path. It can be observed that as soon as the used path violates the policy, experiencing a delay above the threshold, the negative reward triggers the agent to switch path. It can be also seen that the latency of the used path does not increase significantly after F1 traffic has been rerouted on it.

2) *TCP:* TCP performance can be severely impacted by rerouting, causing delay variance, interpreted as losses by congestion control algorithms. To study this problem we performed also experimentation with Cubic TCP. Results in Fig. 11 (middle plot) show that the latency obtained by TCP is comparable with the one obtained by UDP. However, the outliers reach higher values for TCP and there is also a higher jitter (Fig. 11, top plot). This is because TCP retransmits packets that are not received, increasing their delay. We specifically observe retransmissions as spikes in the instantaneous throughput right after the path is switched. The corresponding packets have a higher delay and cause a higher jitter. We also observe that path switching happens before congestion becomes severe, thus preventing TCP flows from experiencing degraded conditions. Fig. 12 depicts the service uptime obtained and the lines on top represent the deviation from a predefined threshold. Generally, TCP flow demonstrates higher compliance with the policy, particularly in highly loaded paths,



(a) Policy A



(b) Policy B

Fig. 10: Instantaneous latency experienced on the alternative paths and on the used path.

as a result of its congestion control mechanism that reduces the sending rate in response to congestion. However, when the threshold is violated, TCP shows a more severe deviation from the threshold owing to the time required for retransmission. The low service uptime obtained in the first experiment is likely a result of the higher number of reconfigurations, which disproportionately impacts the performance of TCP in comparison to UDP.

C. *Policy B results*

In order to evaluate policy B, it is assumed that path 3 simulates a Quality of Service (QoS) guaranteed path, such as MPLS, that imposes a cost per traffic volume that is x times higher in comparison to the other two paths. Path 1 and path 2 simulate best-effort paths, where traffic is generated in a manner such that F2 and F3 overload path 1 and path 2 during the on-times, while F4 never causes congestion on path 3, resulting in delay experienced on this path being always below the threshold. When selecting policy A, the agent would always route F1 to the QoS guaranteed path, as it only considers latency to compute rewards. In contrast, policy B penalizes the use of path 3 due to its higher cost, using it only when the other paths are experiencing poor conditions (Fig. 10b).

A first experiment is conducted for a medium load. By selecting policy B, the agent routes F1 to path 1 or path 2 for 70% of the time, while it uses path 3 for only 30% of the time, despite the fact that path 3 always experiences latency values below the threshold. Even though the paths without QoS guarantees are used more frequently, F1 always experiences low delay by exploiting F2 and F3 off-times (as shown in Fig. 10b). With policy A, instead, path 1 is always used, since its sole aim is to keep latency under the threshold.

The experiment is repeated for various traffic loads and costs. Fig. 13 illustrates the monetary gain of using policy B in comparison to policy A. It can be observed that, as the cost difference between the two paths increases, the gain of policy B also increases. Additionally, the gain rises in low load conditions when the costly path can be used less frequently, since we hypothesize that cost is a function of traffic transiting the path.

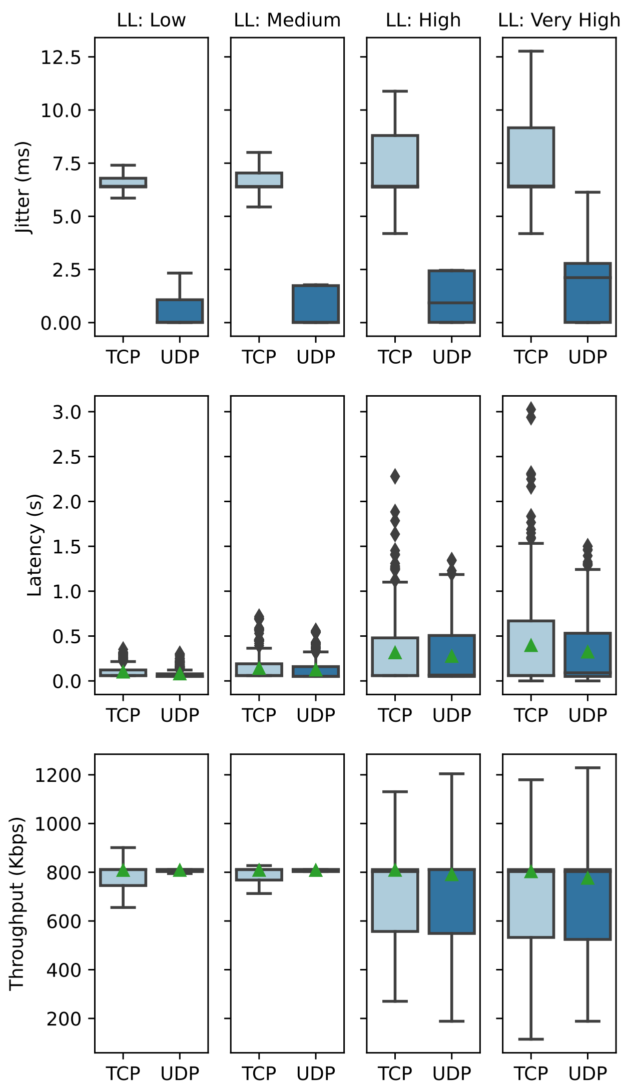


Fig. 11: Network metrics for TCP and UDP using MARL

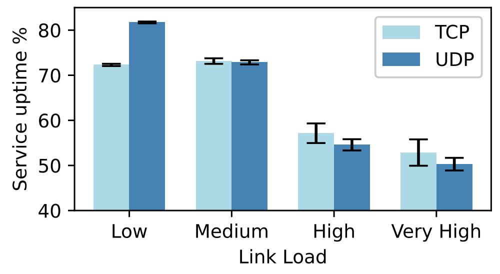


Fig. 12: Service uptime for TCP and UDP traffic using MARL.

VII. Conclusion and Future Work

SD-WAN is considered a promising solution for communication needs of several branch sites located in a wide geographical region connected through different links. In this paper, we started from the observation that Reinforcement Learning has the potential to properly use the available links

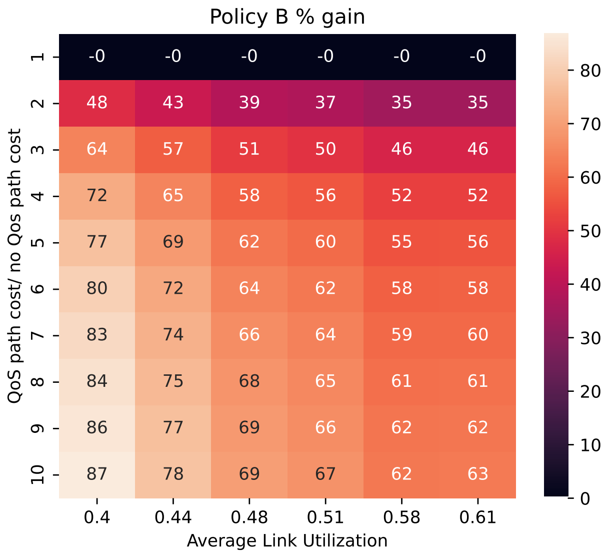


Fig. 13: Policy B gain in relation to different paths costs and increasing loads.

respecting application requirements. We firstly examined the scalability issues of such an approach and proposed MARL as a solution. We showed that this solution effectively mitigates complexity by reducing action and state spaces, utilizing multiple agents that make independent decisions based on local observations of the environment. We then evaluated the effectiveness of this approach in an emulated scenario with multiple sites and overlays. Results show that our approach manages to meet the requirements imposed through the network policy, reducing latency and transit costs, using only local information about the environment. We believe that this paper provides important results that show if and how it is possible to use RL for SD-WAN. Our future work is concerned with the optimization of other parameters of interest such as throughput and available bandwidth.

Acknowledgment

This work was partially supported by Cisco Systems through the Sponsored Research Agreement "Research Project for Industry 4.0" and by the European Union through the ADAPTO project, part of the RESTART program, NextGenerationEU PNRR, CUP E63C22002040007, CP PE0000001.

References

[1] R. Bless, B. Bloessl, M. Hollick, M. Corici, H. Karl, D. Krummacker, D. Lindenschmitt, H. D. Schotten, and L. Wimmer, "Dynamic network (re-) configuration across time, scope, and structure," in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2022, pp. 547–552.

[2] Z. Yang, Y. Cui, B. Li, Y. Liu, and Y. Xu, "Software-Defined Wide Area Network (SD-WAN): Architecture, Advances and Opportunities," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–9.

[3] R. Mohammadi, S. Akleylek, A. Ghaffari, and A. Shirmarz, "Taxonomy of traffic engineering mechanisms in software-defined networks: a survey," *Telecommunication Systems*, pp. 1–28, 2022.

[4] D. Espinel Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 256–281, 2021.

[5] D. Z. Tootaghaj, F. Ahmed, P. Sharma, and M. Yannakakis, "Homa: An efficient topology and route management approach in SD-WAN overlays," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2351–2360.

[6] Y. Xiao, J. Liu, J. Wu, and N. Ansari, "Leveraging deep reinforcement learning for traffic engineering: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2064–2097, 2021.

[7] Enel. (2020) Enel beyond the Cloud: more than a thousand sites connected by one of the world's largest network virtualisation projects. [Online]. Available: https://www.enel.com/media/explore/search-press-releases/press/2020/07/enel-beyond-the-cloud-more-than-a-thousand-sites-connected-by-one-of-the-worlds-largest-network-virtualisation-projects

[8] T. Li, K. Zhu, N. C. Luong, D. Niyato, Q. Wu, Y. Zhang, and B. Chen, "Applications of Multi-Agent Reinforcement Learning in Future Internet: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, 2022.

[9] S. Troia, M. Mazzara, M. Savi, L. M. M. Zorello, and G. Maier, "Resilience of Delay-Sensitive Services With Transport-Layer Monitoring in SD-WAN," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2652–2663, 2022.

[10] P. T. Anh Quang, S. Martin, J. Leguay, X. Gong, and X. Huiying, "Intent-Based Routing Policy Optimization in SD-WAN," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 4914–4919.

[11] Y. Zhang, J. Tourrilhes, Z.-L. Zhang, and P. Sharma, "Improving sd-wan resilience: From vertical handoff to wan-aware MPTCP," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 347–361, 2021.

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

[13] A. Y. Kamri, P. T. A. Quang, N. Huin, and J. Leguay, "Constrained Policy Optimization for Load Balancing," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2021, pp. 1–6.

[14] H. Fawaz, J. Lesca, P. T. A. Quang, J. Leguay, D. Zeghlache, and P. Medagliani, "Graph Convolutional Reinforcement Learning for Collaborative Queuing Agents," *arXiv preprint arXiv:2205.12009*, 2022.

[15] A. Botta, R. Canonico, A. Navarro, S. Ruggiero, and G. Ventre, "AI-enabled SD-WAN: the case of Reinforcement Learning," in *2022 IEEE Latin-American Conference on Communications (LATINCOM)*, 2022, pp. 1–6.

[16] S. Troia, L. M. Moreira Zorello, and G. Maier, "SD-WAN: how the control of the network can be shifted from core to edge," in *2021 International Conference on Optical Network Design and Modeling (ONDM)*, 2021, pp. 1–3.

[17] C. Scarpitta, G. Sidoretti, A. Mayer, S. Salsano, A. Abdelsalam, and C. Filsfils, "High Performance Delay Monitoring for SRv6 Based SD-WANs," *arXiv preprint arXiv:2212.12627*, 2022.

[18] R. K. Rangan, "Trends in SD-WAN and SDN," *CSI Transactions on ICT*, vol. 8, no. 1, pp. 21–27, 2020.

[19] S. Xu, M. Kodialam, T. Lakshman, and S. S. Panwar, "Tomography based learning for load distribution through opaque networks," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 656–670, 2021.

[20] P. Megyesi, A. Botta, G. Aceto, A. Pescapè, and S. Molnár, "Available bandwidth measurement in software defined networks," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ser. SAC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 651–657. [Online]. Available: https://doi.org/10.1145/2851613.2851727

[21] Z. Duliński, R. Stankiewicz, G. Rzym, and P. Wydrych, "Dynamic traffic management for SD-WAN inter-cloud communication," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1335–1351, 2020.