Hierarchical Multi-Agent RL TE in SD-WAN based Cloud Edge Continuum Interconnection

Ayoub Mokhtari
EURECOM
Sophia Antipolis, France
mokhtari@eurecom.fr

Adlen Ksentini
EURECOM
Sophia Antipolis, France
ksentini@eurecom.fr

***Abstract*—The emergence of IoT and the rise of latency-sensitive services are pushing computation from centralised clouds toward a Cloud-Edge Continuum (CEC), where services of the same application are deployed across different CEC nodes, with services requiring low latency closer to the user. This transition introduces new challenges, especially interconnecting these CEC nodes and deployed services dynamically to meet their service level agreements (SLAs). In this paper, we address dynamic Traffic Engineering (TE) in SD-WAN in order to maximise services' QoS requirements fulfilment. We introduce a solution based on Hierarchical Multi-Agent Reinforcement Learning (H-MARL) for dynamically routing service flows through appropriate overlay links that maximise the QoS and reduce the network cost. Simulation results show the efficiency of the proposed solution in dropping overall latency and improving QoS fulfilment by about 10% compared with a flat single-layer MARL.**

***Index Terms*—SD-WAN, Cloud-Edge Continuum, Traffic Engineering, Reinforcement Learning, Hierarchical MARL.**

I. Introduction

The evolution of Cloud computing has transitioned from centralized computing architectures towards distributed paradigms encompassing the Cloud-Edge Continuum (CEC). This paradigm shift responds to the emerging and growing of Internet-of-Things (IoT) devices and latency-sensitive applications, necessitating the deployment of application services closer to the end-user, thus geographically dispersing services across computing nodes. In the CEC model, different services of an application, typically developed as microservices or Cloud-native services, are dynamically placed according to their Quality of Service (QoS) requirements, such as latency, jitter, and bandwidth constraints. For example, critical, latency-sensitive microservices need to be deployed closer to end-users or IoT devices at the Edge or Far Edge nodes, whereas less latency-critical services may remain hosted centrally in Cloud data centers.

Given this distributed deployment, the network interconnection between various nodes in the Cloud-Edge Continuum becomes critically important. Such interconnections must dynamically ensure QoS requirements and Service Level Agreements (SLAs), while managing network resources efficiently and adapting to varying network conditions. Unlike centralized Cloud architectures, where providers often fully control their own network infrastructure, CEC environments typically involve resources spanning multiple administrative domains interconnected via third-party Internet Service Providers (ISPs) (Figure 1). Consequently, direct control over the underlay network is rarely possible, necessitating solutions like Software-Defined Wide Area Networks (SD-WAN) for the heterogeneous multiple public WAN interconnection alternative, and to enable a QoS-driven connectivity.

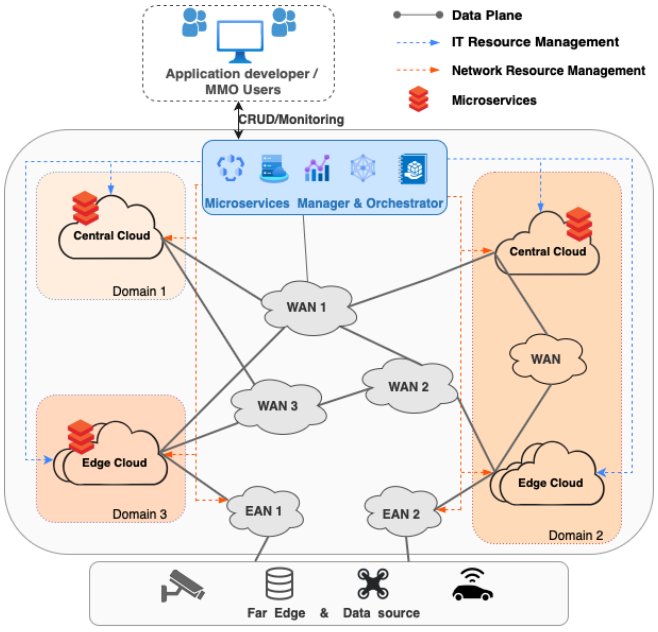


Fig. 1: CEC High-level Framework Overview

SD-WAN emerges as an effective technology for interconnecting CEC nodes by creating overlay networks over heterogeneous underlay infrastructures [1]. Leveraging Software-Defined Networking (SDN) principles, SD-WAN decouples network hardware (data plane) from its control logic, allowing a global overview of the network, dynamic, policy-driven, and application-aware routing. Through such overlay networks, SD-WAN provides programmability, QoS enforcement, and resiliency by dynamically selecting optimal paths based on real-time network conditions, thus supporting the strong demands of distributed, latency-sensitive services in the CEC context [2].

While traditional Traffic Engineering (TE) approaches in

SD-WAN can rely on static policies or heuristic-based routing decisions, they struggle to effectively adapt in real-time to unpredictable network fluctuations. Thus, recent advancements have explored Machine Learning (ML), particularly Reinforcement Learning (RL), to dynamically adapt network routes based on ongoing performance observations [3]. However, single-agent RL approaches face scalability limitations in large-scale networks.

To address these challenges, this paper proposes a Hierarchical Multi-Agent Reinforcement Learning (H-MARL) solution for dynamic Traffic Engineering in SD-WAN based Cloud-Edge Continuum interconnections. The proposed hierarchical design has two levels. A global agent responsible for high-level coordination and resource allocation for instance the allowed link bandwidth to be used, while a local agent at every CEC node routes its own flows, selecting paths that satisfy their QoS requirements. This hierarchical decomposition reduces computational complexity, enhances scalability, and improves overall QoS fulfilment by effectively balancing network cost and application performance requirements.

The remainder of this paper is organized as follows: Section II presents related works in the field of SD-WAN and RL-based Traffic Engineering. Section III describes the problem formulation and mathematical model underlying our approach. Section IV introduces our proposed hierarchical MARL solution, detailing the global and local agent designs. Simulation experiments and performance evaluation are presented in Section V, demonstrating the advantages of our approach over traditional solutions and existing MARL methods. Finally, Section VI concludes the paper and outlines future research directions.

II. Related works

Several works have extensively explored Traffic Engineering (TE) methods leveraging Software-Defined Wide Area Networks (SD-WAN) and Reinforcement Learning (RL) techniques. Troia et al. [4] introduced a monitoring and alert-based overlay selection system in SD-WAN environments to enhance resilience and minimize delays for delay-sensitive services. This system dynamically selects overlays upon detecting performance degradation, validating its approach through real-world and emulated testbeds.

Anh Quang et al. [5] developed a semi-distributed, intent-based routing system aimed at optimizing network overlays to meet specified network objectives, including latency, congestion, and cost minimization. This solution employs local search algorithms combined with traffic prediction models, effectively balancing QoS requirements with operational costs. Similarly, Kamri et al. [6] utilized Deep Reinforcement Learning (DRL) to dynamically determine flow-splitting ratios across multiple network paths, successfully minimizing delays and preventing capacity violations within an SD-WAN infrastructure integrating MPLS and Internet overlays.

In Grandet [7], authors propose a predictive, neural network-driven scheduling framework leveraging Lyapunov optimization to balance QoS demands against transit costs, achieving improvements over static baselines. Dulinski et al. [8] also explored cost-effective traffic management through Dynamic Traffic Management (DTM), designed specifically to minimize inter-domain traffic expenses by dynamically reallocating flows based on monetary and resource-utilization metrics.

Further contributions to SD-WAN optimization include Zhang et al. [9], who addressed inefficient network capacity utilization using active tomography and greedy solutions to tackle minimum-cost routing problems. Their work notably considers NP-hard complexity challenges inherent in routing optimization. Additionally, the study by Dulinski et al. [8] on Dynamic Traffic Management strategies significantly reduced routing costs for inter-cloud communication by intelligently managing flow distributions.

Botta et al. [10] introduced an RL-based dynamic routing framework adapting to unpredictable network conditions, effectively balancing performance and cost objectives, resulting in fewer policy violations and reduced operational costs. Extending this concept further, authors in [11] implemented a scalable Multi-Agent RL (MARL) solution that notably improved service availability and latency reduction in large-scale SD-WAN deployments.

Google's approach, known as B4 [12], demonstrated the potential SDN-based TE to optimize their internal fully controller WAN connecting their data centers. By dynamically allocating competing services' bandwidth based on application priorities, they aim to optimise their WAN link utilisation to nearly 100%. Results of their build TE application showed 2 to 3 times efficiency improvement that enables cost-effective WAN bandwidth utilisation compared to standard practice. Microsoft's SWAN [13] showed how to enhance network utilization through frequent and centrally controlled data plane reconfigurations to match the current network needs. SWAN differentiated traffic into priority classes (interactive, elastic, and background), carefully balancing QoS objectives with fairness considerations. By leaving a small free capacity on links about 10%, they demonstrated a congestion-free reconfiguration and improvement of network performance.

In related network environments, Deep Reinforcement Learning (DRL) techniques have been successfully applied to complex routing and load balancing challenges. Suárez-Varela et al. [14] utilized DRL to optimize automatic routing in optical transport networks. Their DRL agent selects optimal paths from candidate routes to maximize network utilization and accommodate new traffic demands effectively. Similarly, Doke et al. [15] leveraged DRL to design a dynamic load balancer within inter-data center environments. Their approach continuously updates policies in response to evolving network conditions, outperforming traditional deterministic policies such as Round Robin, demonstrating robust adaptability and comparable or superior performance in dynamic settings.

The above works have tackled SD-WAN TE challenges; however, some assumed control over the underlying network, limiting their applicability primarily to private WAN infrastructures. Others, although considering heterogeneous WAN infrastructures, focused only on specific QoS metrics

or neglected scalability, a critical aspect in the federated CEC framework, where nodes can join or leave the federation. In this work, we propose a Hierarchical Multi-Agent RL (H-MARL) solution explicitly designed to address these TE challenges, emphasizing dynamic, per-flow QoS-aware routing across diverse overlay networks within federated and multi-domain Cloud-Edge Continuum environments where direct network infrastructure control is not possible and scalability should be considered.

III. Problem Description and Formulation

A. Description

Figure 1 shows a typical Cloud Edge Continuum nodes interconnection leveraging SD-WAN, where n computing nodes are interconnected with w possible underlying WAN infrastructures potentially administered by different ISP. We consider a set of k overlay networks that consists of SD-WAN edge routers or Gateways (EGs) and peer-to-peer (p2p) virtual tunnels connecting each EG over a given underlay WAN infrastructure. While multiple overlays can be created, we abstract to their underlay infrastructure provider (e.g., MPLS, LTE, Internet). These overlay tunnels are created between SD-WAN Edge Gateways that are deployed at the border of each computing node of the continuum. Leveraging SD-WAN we aim not only to interconnect CEC nodes, but also to maximise the SLA fulfilment of services distributed across CEC by optimizing their flow routing to meet their QoS $\mathbf{Q}_f = (l_f^{\max}, j_f^{\max}, b_f^{\min})$ max accepted latency, jitter and min required throughput.) while minimizing the network costs.

Although, Cloud operators can have the capability when interconnecting their own Data Centers to control the underlay network to specify the level of QoS and path selection for each deployed service function chain (e.g. involve building Segment Routing (SRv6) routes or MPLS tunnels), such an approach cannot be envisioned in a federated cloud edge continuum resource where each CEC node or provider have its ISPs. Thus, it is not possible to control the intra-domain routing, and the only control is at the EGs level. Controlled by the SD-WAN controller, EGs can perform per-packet or per-flow routing through available overlay tunnels, load-balancing, and dynamic path selection for flow QoS enforcement.

In this work, we are interested in dynamic per-flow routing of the continuum services by reacting to the unpredictable nature of the intra-domain network and rerouting the flows to maximise services QoS fulfilment and minimise network cost. To react to the interconnecting underlay network performance degradation, a reliable, nearly real-time monitoring module that delivers metrics of each overlay tunnel (per WAN type) is required. At the learning and simulation phase, we assume having such a module for collecting network performance metrics $\mathbf{M}(e_k) = (l(e_k), j(e_k), b(e_k))$ (measured latency, jitter, and throughput on link e_k). We present in the following subsections the mathematical formulation of this optimization problem.

B. Problem formulation

Symbol	Description
$F = \{f_1, \ldots, f_i\}$	Set of flows.
$E = \{e_1, \ldots, e_k\}$	Set of SD-WAN overlay links.
$Q_f = (l_f^{\max}, j_f^{\max}, b_f^{\min})$	Flow f QoS requirements: max latency, jitter, min throughput.
$M(e_k) = (l(e_k), j(e_k), b(e_k))$	Current overlay link e_k monitored metrics: latency, jitter, throughput.
$p_L(f,k)$	Latency score for routing f over link e_k.
$p_J(f,k)$	Jitter score for routing f over link e_k.
$p_B(f,k)$	Throughput score for routing f over link e_k.
$y_{f,k} \in \{0,1\}$	Binary variable: 1 if flow f is routed on e_k, else 0.
$B(e_k)$	Capacity of link e_k.
$C(e_k)$	Unit cost of using link e_k.
$\alpha_1, \alpha_2, \alpha_3$	Weight coefficients for latency, jitter, and throughput penalties
λ	Cost weight factor in the objective function

TABLE I: Notations

Referring to the notations presented in Table I, the problem described in the previous section could be formulated as follows:

$$\Phi_f = \sum_{k=1}^{K} y_{f,k} \left[\alpha_1 p_L(f,k) + \alpha_2 p_J(f,k) + \alpha_3 p_B(f,k)\right] \quad (1)$$

$$C_{\text{tot}} = \sum_{f \in F} \sum_{k=1}^{K} y_{f,k}\, C(e_k)\, b_f^{\min} \quad (2)$$

$$\max_{\{y_{f,k}\}} \left[\sum_{f \in F} \Phi_f - \lambda\, C_{\text{tot}} \right] \quad (3)$$

Constraints

$$\sum_{k=1}^{K} y_{f,k} = 1 \quad , \quad f \in F \quad (4)$$

$$\sum_{f \in F} y_{f,k}\, b_f^{\min} \leq B(e_k) \quad , \quad e_k \in E \quad (5)$$

$$y_{f,k} \in \{0,1\} \quad , \quad f \in F, \quad e_k \in E \quad (6)$$

Eq. (1) represents the per-flow QoS satisfaction score, which is the aggregation of each QoS metric's score p_L, p_J and p_B weighted with coefficients α_1, α_2, and α_3, respectively. For a given flow f, the value of Φ_f is influenced only by the chosen overlay link e_k performance selected by the binary routing variable $y_{f,k}$. The different QoS metrics scores are defined as follows:

$$p_L(f,k) = \max\left(0, \frac{l_f^{\max} - l(e_k)}{l_f^{\max}}\right) \quad (7)$$

$$p_J(f,k) = \max\left(0, \frac{j_f^{\max} - j(e_k)}{j_f^{\max}}\right) \quad (8)$$

$$p_B(f,k) = \min\left(1, \frac{b(e_k)}{b_f^{\min}}\right) \tag{9}$$

where $l_f^{\max}$, $j_f^{\max}$, and $b_f^{\min}$ denote the maximum accepted latency, jitter, and the minimum required throughput of flow f, respectively; and $l(e_k)$, $j(e_k)$, and $b(e_k)$ are the measured latency, jitter, and throughput of overlay link e_k, respectively.

Eq. (2) defines the total cost C_{tot} incurred by routing flows over overlay links. $C(e_k)$ is the cost per throughput unit for the underlay WAN on which the overlay link e_k is established.

Eq. (3) is the multi-objective function that maximize flows QoS fulfilment while minimizing the network cost. λ is a cost weight coefficient for balancing QoS fulfilment and network cost minimisation.

Eq. (4) and (6) Constraint for routing each flow f exactly through one overlay link e_k. Eq. (5) is the capacity constraint for each link e_k.

The routing problem described above closely resembles the well-known constrained multi-commodity flow problem. This decision problem is proven to be NP-hard through reductions from classical combinatorial problems, such as the 0–1 knapsack problem. Consequently, our routing optimization problem is also NP-hard due to its discrete routing decisions under QoS and cost constraints. Such problems quickly become computationally infeasible as the size of the network and the number of flows increase, making reinforcement learning (RL) a suitable approach to adapt efficiently to varying traffic and network conditions.

IV. Hierarchical Multi-Agent Reinforcement Learning (H-MARL)

In this section, we introduce our proposed solution, Hierarchical Multi-Agent Reinforcement Learning (H-MARL) for dynamic TE interconnecting CEC nodes using SD-WAN. We first present the H-MARL design and then we give a detailed description of the solution.

A. H-MARL design

The upper layer (L0) reasons on a *global view* containing only (i) the aggregated flows demand matrix $D(i,j)$ received from the management plane (Figure 2), (ii) the current load U_k of each overlay link k, and (iii) an aggregate QoS score Q_{global}. From these it outputs a matrix of throttle factors $\pi_{n,k} \in]0,1]$, one value per node–link pair, that limits the share of residual capacity a node is encouraged to use on each overlay path. The lower layer (L1) is instantiated once per node c and keeps visibility of its *local* flows F_c(flows that have c as source) and the overlay link measured metrics, it aims to maximise per-flow QoS satisfaction Φ_f while accounting for the *cost-scaled* price $\hat{C}_{c,k} = C_k/\pi_{c,k}$ announced by L0. At Every Δt_1 each L1 agent evaluates its routing policy and every Δt_0 L0 agent recomputes the π-matrix.

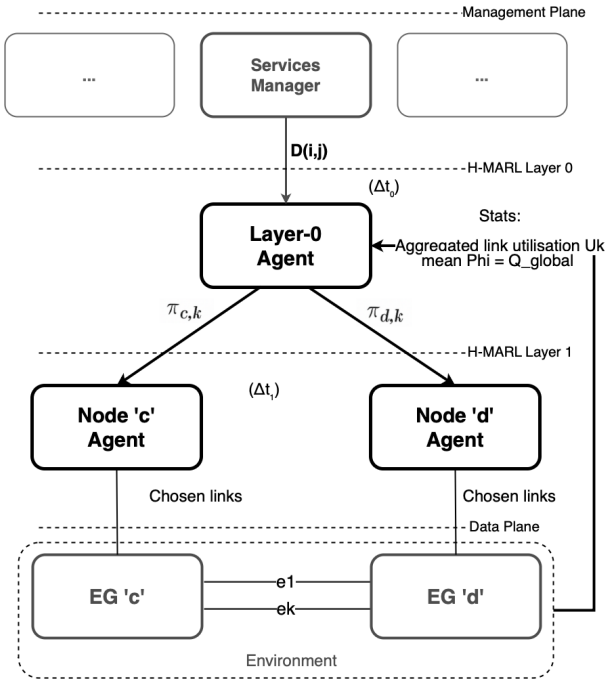


Fig. 2: Proposed H-MARL Solution Design

*1) **Global agent (Layer-0)***

- **State:** The observation of L0 global agent is (i) flows demand matrix $D(i,j) \in \mathbb{R}_{\geq 0}^{n \times n}$ (aggregated minimum-throughput requested by flows from node i to j). (ii) link load $U_k \in [0,1]$ for every overlay link k. (iii) Global QoS score Q_{global} (mean satisfaction $\overline{\Phi_f}$).
- **Action:** The action is $\pi_{n,k}$ which take values between 0 and 1, that defines how a node n may inject at most $\times$(measured available throughput on link k).
- **Reward:**

$$R_0 = Q_{\text{global}} - \gamma \sum_k \max(0, U_k - 1), \quad \gamma \gg 1,$$

where $\gamma \sum_k \max(0, U_k - 1)$ discourages any over utilisation of link k and Q_{global} encourages the overall mean QoS satisfaction. γ is set bigger than 1 so that congestion penalties dominate whenever a link runs above 100 % utilisation, yet any unnecessary throttling that would hurt Q_{global} is avoided because the first term is maximised at the same time.

*2) **Per-node agent (Layer-1)***

- **State:**
  - i) For every local (to node c) flow $f \in F_c$, its QoS requirement $\mathbf{Q}_f = (l_f^{\max}, j_f^{\max}, b_f^{\min})$,
  - ii) For every local overlay link e_k, the latest monitored metrics $M(e_k) = (l(e_k), j(e_k), b(e_k))$,
  - iii) Cost–scaled price $\hat{C}_{c,k} = C_k/\pi_{c,k}$ obtained from the throttle $\pi_{c,k}$ set by L0 global agent, penalising links that tend to congest with a higher cost.
- **Action:** For each active flow f, choose exactly one outgoing link k; this is encoded by a binary variable

Traffic Model	Packet inter-arrival	Packet size (bits)
0	Exponential, $\lambda = 50$ pkts/s	Exponential (mean = $15k$)
1	Truncated Pareto ($\alpha = 1.2$, $m = 3$, max $= 30$)	Truncated Pareto ($\alpha = 1.3$, $m = 20$k, max $= 50$k)
2	Two-state MPP: Exp($\lambda = 10$) / Exp($\lambda = 20$) (pkts/s)	State 1: Uniform $[50k, 70k]$; State 2: Normal ($\mu = 60k$, $\sigma = 5k$)
3	Two-state MPP: Exp($\lambda = 40$) / Exp($\lambda = 80$) (pkts/s)	State 1: Exponential (mean $= 55k$); State 2: Normal ($\mu = 20k$, $\sigma = 6k$)

TABLE II: Background-traffic models and parameters

$y_{f,k} \in \{0, 1\}$ with $\sum_k y_{f,k} = 1$.

- **Reward:**

$$r_c = \sum_{f \in F_c} \Big(\Phi_f \; - \; \lambda \, \hat{C}_{c,k_f} \, b_f^{\min} \Big),$$

where k_f denotes the link chosen for flow f. The first term rewards flows whose latency, jitter and throughput meet their targets, while the second term penalises expensive routing decisions. Because $\hat{C}_{c,k}$ increases when $\pi_{c,k}$ is small, the agent is guided away from links that L0 wants to protect from congestion without the need for hard capacity caps.

B. Detailed description

For the layer 0 agent *Soft Actor–Critic* (SAC) [16] is used as the action space is continuous specifically throttle matrix $\pi_{n,k}$. The global agent acts only once every Δt_0 and therefore has comparatively few samples. For layer 1 agents *MAPPO with parameter sharing* is used with decentralised actors and centralised critic available only during training to improves credit assignment across nodes agents. Parameter sharing is appropriate because every edge node solves the same routing sub-problem, thus using one policy speeds up the learning phase.

Both the actor and critic of *SAC* are implemented as compact two-layer, fully connected (feed-forward) neural networks, with each hidden layer containing 256 `ReLU`-activated units. The actor's final activation is a `tanh`, whose output is linearly rescaled to the interval $]0, 1]$. We retain the standard SAC learning rate of 3×10^{-4}, adopt a target-network update coefficient of $\tau = 0.005$, and maintain a replay buffer of 10^6 transitions. The entropy temperature α is tuned automatically by the built-in update rule, so no manual sweep is required. One gradient step is taken after each environment interaction using a mini-batch of 2 048 samples.

MAPPO actor of a given node consists of two fully connected layers with 128 ReLU units. The centralised critic sees the concatenation of all node observations. PPO clipping is set to 0.2 and GAE parameter to $\lambda = 0.95$.

V. Simulation and Results

In this section, we will introduce simulation environments and traffic generation models (Table II). Then, We will evaluate our proposed solution and present parameters used for training the agents (Table III).

For training, we built a discrete-event simulator in Python using the `simpy` framework [17], following the implementation of the open-source simulator *ns.py* [18]. The simulator reproduces packet-level queuing, per-link delay, and loss.

In order to test with a real SD-WAN solution, we tested overlay link flipping (Figure 6) by using an emulated environment on top of Mininet [19] running our SD-WAN implementation [1] with 3 EGs and an SD-WAN controller with 3 emulated WAN links interconnecting CEC nodes.

Traffic flows are also generated using the discrete event simulator, using different probabilistic distributions to mimic four distinct service classes: voice, video, media streaming, and bulk data transfer. Classes could use fixed or exponential inter-arrival times for real-time streams, exponential or uniform inter-arrival times for video and data transfer traffic, and normal or Pareto packet-size distributions to emulate small-chunk voice, medium-sized video, and bulk data traffic. These configurations provide a realistic mixture of latency-sensitive and throughput-intensive traffic, challenging the routing agents to balance QoS and cost effectively.

We have implemented our solution using the Pytorch library [20]. The different considered parameters for the proposed design agents are presented in Table III. All experiments were conducted on a machine with 20 CPU threads (12th Gen Intel Core i7-12700, Turbo @ 4.9 GHz) and 128 GB RAM.

Parameter	Value
Layer–0 (SAC) agent	
Discount factor γ	0.98
Learning rate (Adam)	3×10^{-4}
Hidden layers (actor & critic)	2×256 ReLU
Replay-buffer size	1×10^6 transitions
Mini-batch size	2 048
Soft-update coefficient τ	0.005
Entropy temperature α	Auto-tuned
Layer–1 (shared MAPPO) agents	
Discount factor γ	0.95
Learning rate (Adam)	2.4×10^{-4}
Hidden layers (actor & critic)	2×128 ReLU
GAE parameter λ	0.95
PPO clip ratio ϵ	0.19
Roll-out length	8 192 time-steps
Mini-batch size	256
Policy updates per roll-out	4

TABLE III: Training parameters used for each H-MARL layer

We have evaluated our proposed solution in terms of: (i) convergence behavior, (ii) QoS fulfillment under varying traffic conditions and network scales, and (iii) latency optimization performance for latency-sensitive flows.

Figure 3 shows how the average normalised reward evolves during training for two algorithms compared to our proposed solution. The first curve, labelled *SARL-L* (PPO–Local), corresponds to a single-agent PPO that can see only the overlay links and flows of one node at a time. Because its state space is small, the algorithm begins to improve before the 2 other and

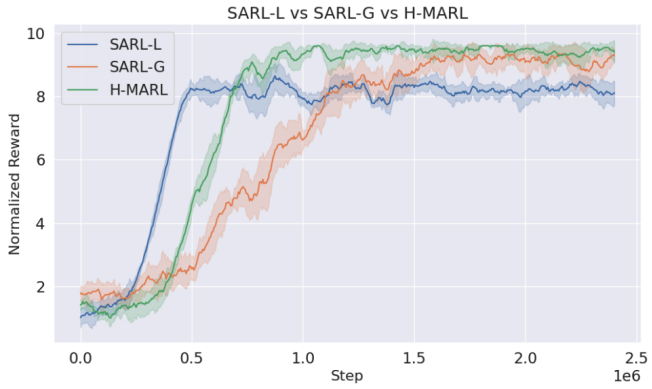


Fig. 3: Compared solutions Normalized Reward convergence

reaches a stable value after roughly 4×10^5 steps. However, it settles at about 8 on the normalised scale and sometimes gets under, which means the policy still leaves noticeable room for optimisation.

A second curve, *SARL-G* (PPO–Global), represents a single PPO agent with full visibility of the flows' QoS demand and every link metric. The wider state space makes learning slower: the reward starts to rise only after an extended exploration phase and does not stabilise until nearly 1×10^6 steps. This result shows that it's not a scalable approach to the problem defined, especially as the number of flows goes higher.

The green curve depicts the proposed hierarchical solution, abbreviated *H-MARL* (SAC+MAPPO). In this setup, a global SAC agent learns throttle factors while node-level routing is handled by shared MAPPO actors. The curve rises more slowly than PPO–Local at first, because the two layers must co-adapt, yet it overtakes the local single-agent RL approach around 7×10^5 steps. It also converges sooner than PPO–Global and finally stabilises at a higher reward, approximately 9.2. These observations indicate that a lightweight global throttle is sufficient to coordinate decentralised node agents effectively to avoid congestion, while avoiding the complexity of a one global policy.

Figure 5 compares the percentage of flows whose QoS targets are met under two policies, two different nodes number ($N = 10$ and $N = 50$) and three background traffic load Low, Medium and High, where parameters for low load are described in table II. The results were obtained comparing our proposed approach labed *SAC + MAPPO (H-MARL)* with two algorithms: (i) the single-agent PPO that observes only a node's local state (PPO-L) and (ii) a flat MAPPO in which one PPO actor runs on every node with parameter sharing. Results are averaged over 25 independent simulation runs, and error bars mark the 95% confidence interval of the mean. The horizontal axis groups the experiments into *Low*, *Medium*, and *High* background-traffic loads, while the vertical axis reports the resulting QoS-fulfilment.

Figures 5.a and 5.b use a cost weight of $\lambda = 0.6$ (see equation 3), giving priority to satisfying flows if the chosen path is not always the cheapest. With ten nodes $N = 10$ in 5.a the proposed hierarchical design reaches around 85% fulfilment under low load and still maintains almost 80% when the background traffic load is high. On the other hand, both baseline algorithms drop in QoS fulfilment with MAPPO losing 5% to 8%, and the single-agent PPO drops below 55% especially with $N = 50$.

Figure 5.c and 5.d show the result of the same experiment but with $\lambda = 1.0$, giving equal weight to network cost and QoS. As expected, QoS fulfilment values of all algorithms decline because the agents now balance two conflicting objectives with equal importance. Even so, our proposed SAC+MAPPO approach demonstrates better results. For ten nodes, it preserves close to 70% fulfilment at low load and stabilises around 55% at high load, outperforming MAPPO and PPO by 8–12%. With $N = 50$ nodes, the overall fulfilment falls further, yet the proposed hierarchical policy shows consistently better results, demonstrating that the global throttle helps the node agents avoid paths that would become congested even when cost is taken into consideration.

Figure 4 summarises aggregated active flows latency comparing the proposed Hierarchical MARL with the other two baseline MAPPO and the single agent PPO-L, when latency is given the highest weight in the QoS score (bigger α_1 in Eq. 1) for latency sensitive services. The aggregated flows latency results are obtained with different values of N (N=10, N=20) which denote the number of nodes and F (F=20, F=50) which denote the number of flows we aggregate their latency for each experiment. The red triangle marks the mean, the box spans the inter-quartile range, and circles denote outliers. Colour shading follows the bar on the right-hand side and indicates the average fraction of link capacity (over all the overlay links) already consumed by background traffic during each experiment.

With $N = 10$ nodes and $F = 20$ flows, all three algorithms achieve comparable medians. As the number of flows grows to 50, we observe the hierarchical MARL approach has better results with a lower median latency, whereas the single-agent PPO shows a long tail that stretches beyond 0.85 s.

Increasing the number of nodes to $N = 20$ magnifies the differences. With 20 active flows, the proposed SAC + MAPPO hierarchy holds the median latency around 0.25 s, whereas both PPO – Local and flat MAPPO show wider spreads and several runs above 1 s. At the highest load ($N = 20$ nodes, $F = 50$ flows), the local PPO shows poor performance with some values going near $1.9s$ and flat MAPPO exceeds sometimes $1.5s$, while the hierarchical MARL has upper values near $1s$. These results indicate that the global SAC agent throttle curbs congestion on busy links and thus helps reduce delay caused by congested links.

Figure 6 illustrates how the trained H-MARL agent reacts to changing network conditions in the emulated SD-WAN environment described above. The experiment uses three SD-WAN edges connected by three emulated underlay networks on top of which we create overlay vxlan tunnels whose one-way delays are plotted over 30 simulation steps (blue,

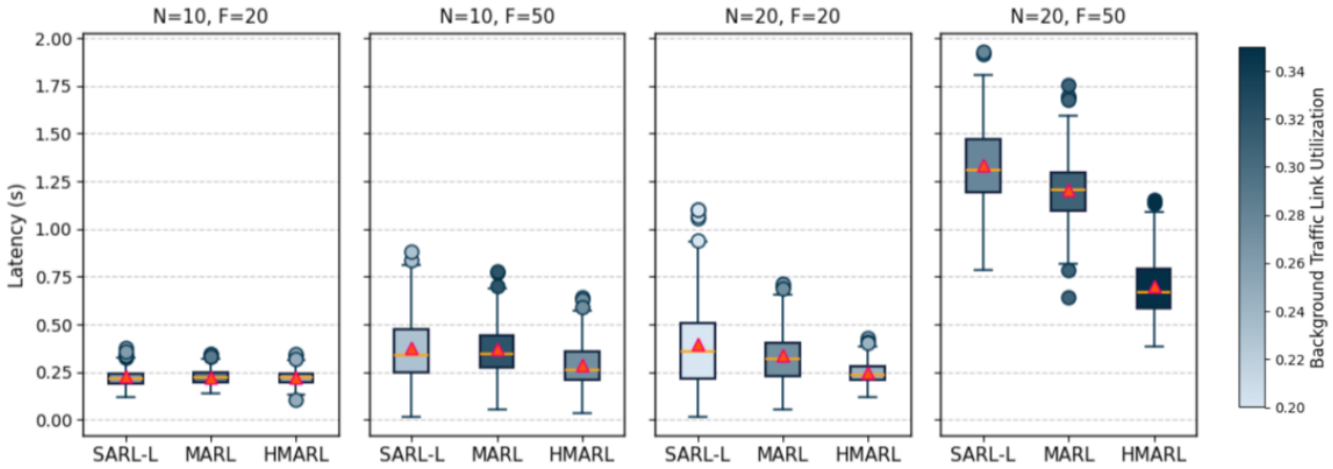
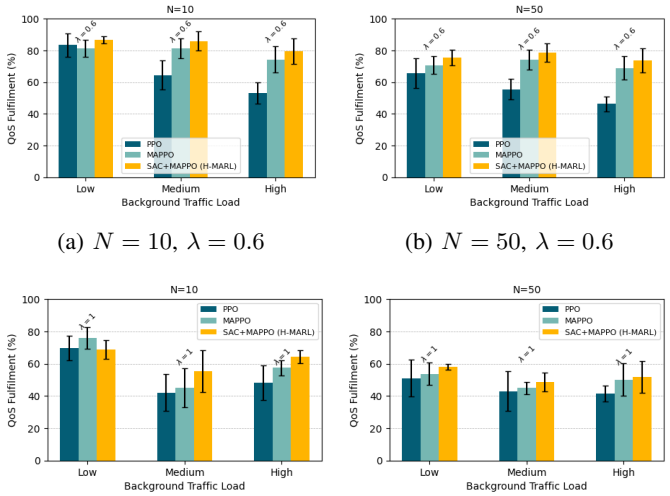


Fig. 4: Services flows overall experienced latency



(a) $N = 10$, $\lambda = 0.6$ (b) $N = 50$, $\lambda = 0.6$

(c) $N = 10$, $\lambda = 1$ (d) $N = 50$, $\lambda = 1$

Fig. 5: Services QoS fulfilment

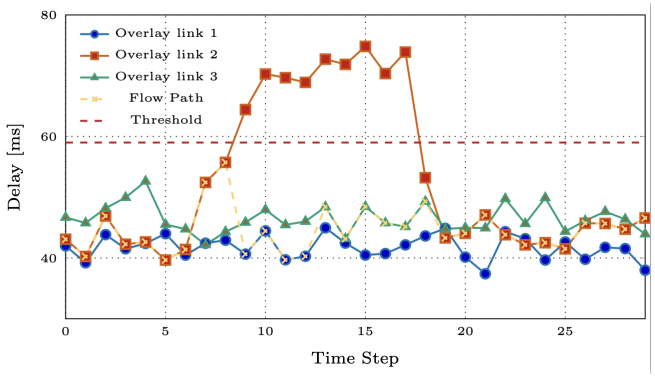


Fig. 6: Dynamic Overlay selection of a given flow

orange and green). The red dashed line marks the QoS latency threshold of 59 ms of a flow, and the yellow dashed line shows the currently selected tunnel for this flow that originates from SD-WAN Edge 1. Background delay on Overlay 2 is increased with `netem` [21] between steps 8 and 18 to emulate sudden congestion.

Until step 7, the flow follows Overlay 2 because it offers the lowest price and its delay remains below the threshold. When the delay of Overlay 2 starts to rise, the local MAPPO router at Edge 1 guided by the global SAC switches the flow to Overlay 1 at step 8, avoiding a QoS requirement violation. At step 13, the flow was shifted to overlay 3, whose delay is still under the threshold, as overlay 1 became less promising after we emulated SD-WAN Edge 2 routing flows through overlay 1. This behaviour is a consequence of the global SAC agent redefining the throttle $\pi_{1,1}$ guiding node 1 agent to avoid congestion by reducing the throughput via link 1.

Finally, after the mulated congestion using *netem*, the was routed back to overlay 2, which is the most cost-effective route for the current flow. These results show that the hierarchical controller can both predict deteriorated delay or spikes and redistribute traffic, while respecting the cost.

VI. Conclusion

In this paper, we proposed a Hierarchical Multi-Agent Reinforcement Learning (H-MARL) solution to address the dynamic Traffic Engineering (TE) challenges in Software-Defined Wide Area Networks (SD-WAN) tailored for Cloud-Edge Continuum (CEC) environments. Unlike traditional single-agent or flat multi-agent solutions, our hierarchical approach effectively balances the complexity and scalability of the optimization problem. The global agent coordinates high-level resource allocation, guiding local agents in making precise, application-aware routing decisions. Simulation results demonstrated that our solution significantly enhances QoS fulfilment, reduces network latency, and maintains cost efficiency compared to existing single-layer MARL and single-agent approaches. Future research directions could be a further

validation of the proposed solution by comparing it with other Hierarchical Multi-Agent Reinforcement Learning (H-MARL) algorithms, exploring more advanced RL methods, and investigating the reliability of the monitoring system in such a decentralized design.

ACKNOWLEDGMENT

This work is partially supported by the European Union's Horizon Program under the 6G-Intense projects (Grant No. 101139266).

REFERENCES

[1] A. Mokhtari and A. Ksentini, "Sd-wan for cloud edge computing continuum interconnection," in *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, 2024, pp. 2533–2538.

[2] A. Navarro, R. Canonico, and A. Botta, "Software defined wide area networks: Current challenges and future perspectives," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*. IEEE, 2023, pp. 350–353.

[3] 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)*. IEEE, 2019, pp. 1–9.

[4] 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.

[5] P. T. A. 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*. IEEE, 2022, pp. 4914–4919.

[6] 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.

[7] Y. Zhang, H. Zhang, P. Cong, W. Wang, and K. Xu, "Grandet: cost-aware traffic scheduling without prior knowledge in sd-wan," in *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE, 2023, pp. 1–10.

[8] 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.

[9] 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.

[10] 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)*. IEEE, 2022, pp. 1–6.

[11] A. Botta, R. Canonico, A. Navarro, G. Stanco, and G. Ventre, "Scalable reinforcement learning for dynamic overlay selection in sd-wans," in *2023 IFIP Networking Conference (IFIP Networking)*. IEEE, 2023, pp. 1–9.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[13] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 15–26.

[14] J. Suárez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routing based on deep reinforcement learning in optical transport networks," in *Optical Fiber Communication Conference*. Optica Publishing Group, 2019, pp. M2A–6.

[15] A. R. Doke and K. Sangeeta, "Deep reinforcement learning based load balancing policy for balancing network traffic in datacenter environment," in *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE, 2018, pp. 1–5.

[16] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[17] "Simpy documentation," https://simpy.readthedocs.io, 2025, accessed: 2025-07-02.

[18] B. Li, L. Chen, and X. Peng, "ns.py: A pythonic discrete event network simulator," https://github.com/TL-System/ns.py, 2022, accessed: 2025-07-02.

[19] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[20] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "Pytorch," *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104, 2021.

[21] "tc-netem(8) - traffic control network emulator," https://man7.org/linux/man-pages/man8/tc-netem.8.html, accessed: 2025-07-02.