Autonomous Flow Routing for Near Real-Time Quality of Service Assurance

Sima Barzegar, Marc Ruiz, and Luis Velasco

***Abstract*—The deployment of beyond 5G and 6G network infrastructures will enable highly dynamic services requiring stringent Quality of Service (QoS). Supporting such combinations in today's transport networks will require high flexibility and automation to operate near real-time and reduce overprovisioning. Many solutions for autonomous network operation based on Machine Learning require a global network view, and thus need to be deployed at the Software-Defined Networking (SDN) controller. In consequence, these solutions require implementing control loops, where algorithms running in the controller use telemetry measurements collected at the data plane to make decisions that need to be applied at the data plane. Such control loops fit well for provisioning and failure management purposes, but not for near real-time operation because of their long response times. In this paper, we propose a distributed approach for autonomous near-real-time flow routing with QoS assurance. Our solution brings intelligence closer to the data plane to reduce response times; it is based on the combined application of Deep Reinforcement Learning (DRL) and Multi-Agent Systems (MAS) to create a distributed collaborative network control plane. Node agents ensure QoS of traffic flows, specifically end-to-end delay, while minimizing routing costs by making distributed routing decisions. Algorithms in the centralized network controller provide the agents with the set of routes that can be used for each traffic flow and give freedom to the agents to use them during operation. Results show that the proposed solution is able to ensure end-to-end delay under the desired maximum and greatly reduce routing costs. This performance is achieved in dynamic scenarios without previous knowledge of the traffic profile or the background traffic, for single domain and multidomain networks.**

***Index Terms*—Quality of Service Assurance, Flow Routing, Network Automation, Near-real-time control, Deep Reinforcement Learning, Multi-Agent Systems.**

I. Introduction

TRANSPORT networks need to be redesigned to support the expected large traffic dynamicity and stringent performance of beyond 5G (B5G) and 6G services. Such support requires increased levels of flexibility and automation, together with higher priority given to network optimization and cost efficiency [1]. As a result, solutions for autonomous network operation based on the application of Artificial Intelligence (AI) / Machine Learning (ML) have been proposed in the last years (see, e.g., [2]) to implement data-driven closed control loops. Such solutions running in a centralized element, have the potential to greatly reduce operational costs by minimizing human intervention. Because of its global view, current network architectures rely on the Software Defined Networking (SDN) controller as the ideal component where decision making should be carried out, being this approach beneficial for many applications, like service provisioning, failure management, etc. In consequence, SDN control is being augmented with instantaneous data-driven decision-making [3].

However, precisely because of its centralized location, (near) real-time decision making does not fit well with SDN controllers. In particular, in the case that automation needs to deal with highly dynamic traffic conditions, centralized decision-making leads to poor resource utilization because of long response times. In this work, we focus on flow routing, where decisions need to be made near real-time to optimize resource utilization while ensuring the Quality of Service (QoS) of the flows. Note that traffic variations might create bottlenecks that impact on the end-to-end (e2e) delay, defined as the time required for transmitting flow traffic between the two border packet nodes.

To minimize response time, as well as the amount of data to be conveyed from collection points to decision-making elements, AI/ML algorithms might be executed as close as possible to the data sources (contrarily to the centralized architecture of SDN). A possible solution is to use (Deep (D)) Reinforcement Learning (RL) [4], [5]. (D)RL has been proposed for solving problems that require real-time decision making, like the management of the capacity of packet links [6] or that of optical connections [7].

In our previous work [8], we proposed a distributed autonomous inter-domain flow routing based on DRL running in the packet nodes, following the concept of Multi-Agent Systems (MAS) [9]. MAS is a subfield of AI and it can be defined as a set of individual agents that share knowledge and communicate with each other in order to solve a problem that is beyond the scope of a single agent. In the scope of networking, we proposed that agent nodes make autonomous decisions near-real-time based on guidelines received from the SDN controller, thus liberating the SDN controller from near-real-time operations. That system autonomously routes

The research leading to these results has received funding from the European Commission thought the HORIZON SNS JU DESIRE6G (G.A. 101096466), the MINECO UNICO5G TIMING (TSI-063000-2021-145) and the MICINN IBON (PID2020-114135RB-I00) projects, and from the ICREA Institution.

Sima Barzegar, Marc Ruiz, and Luis Velasco (luis.velasco@upc.edu) are with the Optical Communications Group at Universitat Politècnica de Catalunya, Barcelona, Spain.

packet flows entering in the node considering the measured e2e delay, but without previous knowledge of traffic characteristics because of its ability to learn.

The rest of the paper is organized as follows. Section II reviews the state of the art and summarizes the contributions of our work. Section III sketches the main concepts and challenging scenarios of the proposed DRL-based flow routing operation. Section IV details the flow routing agent, including the formal definition of the DRL engine, as well as the algorithms and procedures involved during provisioning and operation under the considered scenarios. Section V presents numerical results obtained by simulation to validate the aforementioned contributions. Finally, Section VI draws the main conclusions.

II. Related work and Contributions

The Open Shortest Path First (OSPF) protocol has been commonly used for intra-domain routing. Under OSPF, packets are routed through one or multiple equal-cost shortest paths (ECMP) to the destination, computed based on links weights that are set inversely proportional to links capacity [10]. Although OSPF works well for optimal general routing, links weights are static, so OSPF cannot avoid congestion during peak traffic under dynamic traffic scenarios. Many works have proposed solutions for Traffic Engineering (TE) that optimize link weights for minimizing network congestion, e.g., based on heuristics, as in [11]. Targeting at significantly lower execution times, which is of paramount importance in the case of highly dynamic traffic scenarios, other works have proposed ML techniques for TE optimization, like [12] and [13], where the use of DRL was proposed, and [14] that leverages MAS for distributed TE optimization. Multiprotocol Label Switching (MPLS) is another routing technology that can be combined with TE for traffic steering [16].

In the packet network, SDN provides a centralized flow control and can guarantee per-flow performance. SDN defines flow rules in the packet nodes that are applied to the incoming packets. Many approaches for flow routing in the centralized SDN controller can be found in the literature (see the survey in [15]). E.g., the authors in [17] proposed a RL-based routing algorithm for TE running on top of the SDN controller considering throughput and delay. In addition, the SDN approach can be used for developing solutions to ensure per-flow performance, as in [18], where the authors focused on guaranteeing per-flow throughput and packet loss. Finally, regarding routing in multi-domain scenarios, the authors in [19] proposed a RL-based QoS-aware adaptive routing that works on top of a hierarchy of network controllers.

Looking at B5G/6G networks, they are foreseen to be not only very dynamic, but also services will require stringent performance. A specific technique that can be implemented to facilitate meeting target performance of traffic flows is multipath, which can be realized by the end hosts at the Transmission Control Protocol (TCP) level, where several sub-flows are established to provide better resource utilization and increased throughput [20]. Intelligent schedulers assign packets to sub-flows based on the measured round-trip-time and throughput of each path (see e.g., the work in [21] based on DRL). Such strategy has been also explored for traffic flows. For instance, the authors in [22] proposed a RL-based for packet routing to optimize path cost and e2e delay, assuming a global view of the network and traffic matrix to decide global routing strategies. A load balancing policy was defined to characterize the distribution of the traffic over the best *n* paths. Some other works have targeted at controlling the traffic split ratio to multiple paths on top of the SDN controller (see, e.g., [23] and [24]). However, routing large flows over several paths might create packet out of order problems. To mitigate such negative impact, the authors in [25] proposed a RL-based scheme that routes the majority of the traffic flows using ECMP, selects some critical flows from the traffic matrix and reroute them using SDN to balance link utilization.

Our approach is also based on the use of several paths to ensure per-flow QoS (specifically e2e delay) and assume the flows to be splitable, i.e., they consist of a large number of sub-flows, which can be routed independently. However, our proposal -also based on SDN- makes decisions in a distributed manner, which has benefits under highly dynamic scenarios where near-real-time routing decisions need to be made. Node agents make routing decisions based on the measured QoS of the paths that have been computed by the SDN controller. Specifically, in this paper, we extend our initial work in [8] and propose a comprehensive solution for near real-time DRL-based flow routing built on the principles of MAS operation. Two phases are considered: *i*) during flow *provisioning* phase, the SDN controller decides which routes (including the case where a single one is computed) can be used for the traffic of the flow and gives degrees of freedom to the agents to use one or a combination of them at any given time. Then, the flow rules enforcing those routes are installed in the involved nodes, so the packet node can forward any incoming packet that matches the rule. The source node receives also the performance required for the flow, e.g., the maximum e2e delay; and *ii*) during the *flow operation* phase, the agent at the source node decides dynamically the proportion of traffic of the flow that is sent through each of the interfaces that belong to the routes received from the SDN controller, targeting at meeting the target performance. Note that any other intermediate node will apply the rules already installed to the incoming packets. The agent at the source node changes the traffic proportion after receiving measurements of the achieved performance without further iterations with the SDN controller. In the case where the agent cannot meet the target performance, it will issue a notification to the SDN controller, so the latter can change the set of routes to be used for routing the traffic of the flow. As a result, the proposed architecture will relieve the SDN control plane from near real time operation and now focus on those activities that require long-term analysis and/or are hard to implement distributedly, like failure management [26]. Specifically, if the SDN controller needs to *reconfigure* the network, it can change the set of routes that can be used for every flow.

A key novel contribution of this paper is the design and evaluation of the DRL-based flow routing agent, responsible of managing the routing of a target flow. From the lessons learnt in [6], a number of modules and workflows are presented to allow efficient and robust DRL-based routing immediately after the flow is provisioned and operation starts. Note that the DRL engine is able to learn and improve actions as soon as it gets experience from operation. In this regard, the designed flow routing agent is able to learn not only how to better adapt flow routing to traffic changes but also, to understand how network state impacts on the QoS assurance of that flow. In particular, we propose solutions to deal with the presence of background traffic competing with the target flow for common capacity resources. We demonstrate that such learning works effectively without any measurement of the background traffic, even though it changes over time.

Finally, we face multi-domain scenarios, where domains coordinate to assure the maximum e2e delay by ensuring per-domain delay budgets. Due to congestion in one or more domains, the delay budget in others might need to be reduced to achieve the target e2e delay. Therefore, the proposed DRL-based flow routing agent is designed to adapt to sudden and asynchronous changes in the delay requirement for the domain, while keeping robustness of operation. Inter-domain interfaces are proposed for the domains to share delay budgets among them.

III. Distributed Network Intelligence For Autonomous Flow Routing

In this section, we first present the high-level architecture of the proposed solution for autonomous flow routing with QoS (delay) requirements. The delay experienced by a given traffic flow depends on the actual traffic volume not only of that traffic flow, but also of the other flows that use common packet links. Two cases are analyzed considering single domain and multi-domain scenarios.

Let us first focus on the single domain. In a pure SDN-based approach, packets in a flow follow the route computed by the SDN controller. Route selection can be performed at flow *provisioning* time based on the network topology and current network conditions. Let us assume that the SDN controller computes a set of allowable routes at packet flow provisioning time. Each route needs to meet the required maximum e2e delay ($dmax_{e2e}$) considering only static delay components, such as *transmission* and *processing delay*, as well as some traffic volume for that flow and other flows currently established in the network. A possible routing policy would be to use the route with the minimum delay for the packet flow. Once a route is selected, it does not usually change until some event occurs. This process requires high overprovisioning, i.e., poor resource utilization, since route selection needs to be performed considering the maximum traffic volume for the packet flows. Otherwise, if volumes exceed the value considered at provisioning time, such policy might result in high delay coming from the time that every packet has to spend in the queues in the packet nodes (*queueing delay*). Hence, reducing overprovisioning while guaranteeing that actual e2e delay (d_{e2e}) for the packet flows do not exceed $dmax_{e2e}$ is a difficult task under dynamic traffic, since every output interface in the packet nodes operates at a different load.

To deal with this, we propose the distributed intelligence architecture sketched in Fig. 1, where a single node and the centralized SDN controller are represented. Note that in such architecture, we are moving the intelligence from the centralized SDN controller to the nodes, thus resulting in a hybrid centralized SDN control with *distributed network intelligence*: *i*) *coordination* among agents is achieved by letting them to communicate with other agents at remote nodes to exchange observed data and/or models; *ii*) *decision-making* is performed by every individual agent near real- time (sub-second to few second granularity) based on its own observed data, as well as on the data and models received from other agents; and *iii*) the SDN controller plane is responsible for the *general coordination* of the network, being in charge of generating the needed *guidelines* for the agents while leaving the desired degree of freedom for their autonomous operation. It is worth noting that this distributed architecture is aligned with current trends on monitoring and data analytics architectures for autonomous networking [3].

Fig. 2 represents an example of flow routing operation in a packet node. The packet node agent has received from the

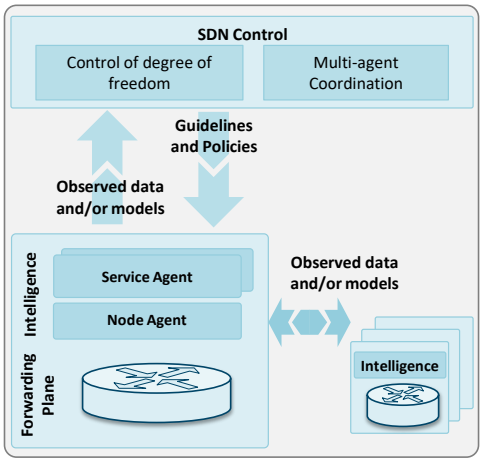

Fig. 1. Proposed distributed intelligence.

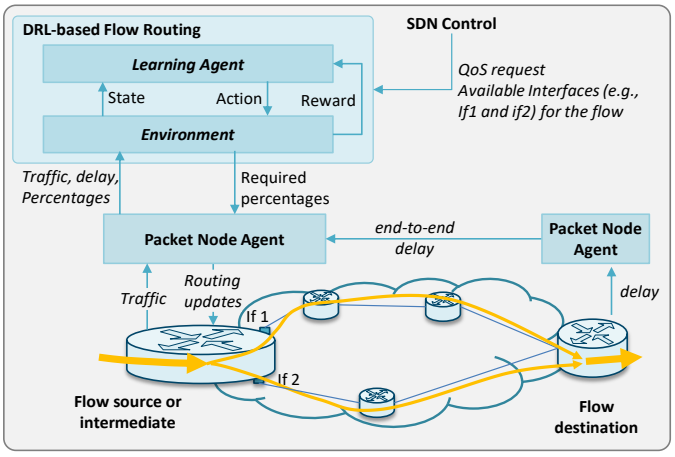

Fig. 2. Example of distributed flow routing based on DRL.

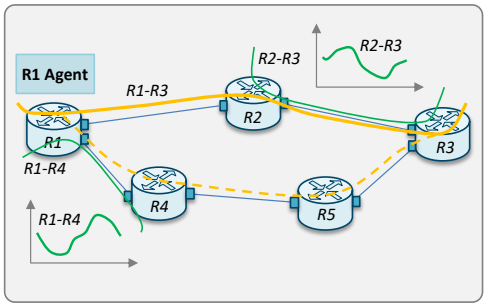

Fig. 3. Example of operation under uncertain background traffic.

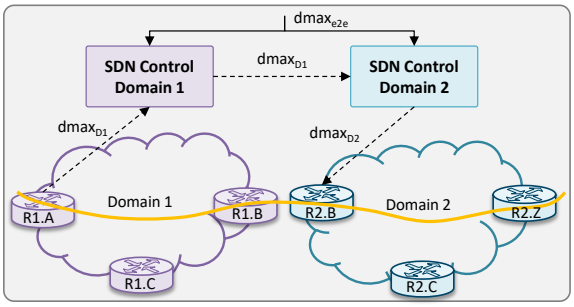

Fig. 4. Example of operation under varying *dmax* in multi-domain scenarios.

SDN controller the required QoS and two possible routes with different utilization cost for a given traffic flow and it has to decide which route or combination of them allow to reach the committed QoS performance (i.e., do not exceed $dmax_{e2e}$), while minimizing some cost function. As represented in Fig. 2, we assume DRL to implement a service agent for the traffic flow. The DRL agent consists of two inter-related blocks: *i*) the *learning agent* is in charge of learning the best actions to be taken based on the current state and the received reward; and *ii*) the *environment* that is in charge of computing the state based on the observed traffic and current combination of routes (proportions), as well as the obtained reward based on the measured e2e delay for the selected routes. Without loss of generality, we assume that the DRL agent operates under the life-cycle presented in [6], where models are pre-trained offline with generic data and refined during online learning, thus achieving both decision-making robustness and high performance from the beginning of flow operation. In Fig. 2, the traffic of the flow is routed through the two different output interfaces, and thus the flow follows different routes. Here, we assume that the traffic flow actually consists of multiple sub-flows, which are routed independently so the packets belonging to each sub-flow follow the same route. In the destination, the e2e delay is measured (e.g., using in-band telemetry) and some statistics are computed (e.g., maximum and weighted average) and sent to the node agents participating in the routing of the flow.

It is worth mentioning that, in this example, the intelligence requirements of the node agent are simple if only the traffic volume of the flow varies with time. Such variation (e.g., typical daily patterns) can be modelled and consequently, predicted with sufficient accuracy. However, let us now present some challenging scenarios where learning is needed due to the variation of the traffic flow under-control and the network dynamicity.

In the first scenario, we deal with changing traffic conditions of other packet flows using some of the links in the set of routes for the traffic flow under control. To illustrate this, Fig. 3 shows an example in a simple network with five packet nodes, where three packet flows are depicted: R1-R3, R1-R4, and R2-R3. Inner graphs in Fig. 3 illustrate possible traffic variation with time for packet flows R1-R4 and R2-R3. Let us assume that R1 has received from the SDN controller two possible routes for flow R1-R3, i.e., R1-R2-R3, with two hops, and R1-R4-R5-R3, with three hops. Initially, the shortest route can be selected and 100% of the traffic in the flow arriving in R1 is routed to R3 through the link R1-R2. Let us imagine that with such configuration R1 receives e2e delay measurements from R3 for flow R1-R3 which are under the given maximum for that flow. At some point in time, flow R2-R3 is established and the e2e delay of flow R1-R3 exceeds the maximum, so R1 needs to route part of the traffic of the flow through the second route that has a higher utilization cost, i.e., through R4. In this way, the traffic through the first route will reduce, which could result in reducing e2e delay. Later, traffic variation in the flows can result in delay variations, which will cause R1 to make new routing decisions, either to reduce delay or to minimize cost.

The second challenging scenario is in multi-domain networks, where a packet flow traverses two different administrative domains. Examples, include access networks (fixed or mobile) and metro/core networks. Although the e2e traffic flow consists of two segments, one in each domain, $dmax_{e2e}$ needs to be ensured. Even when each domain works under low to moderate load regime, delay fluctuations are produced as a result of traffic variations, which makes load also variable in time. In this case, the SDN controllers of each domain have received the required $dmax_{e2e}$ for the traffic flow at provisioning time. It is worth noting that if both domains operate without any coordination among them, large capacity overprovisioning is required to absorb delay variations introduced not only by the own domain, but also by the other domains traversed by the flow. In view of that, we assume some sort of coordination among domains. An example is represented in Fig. 4. The SDN controller of domain 1 dynamically gets the delay that can be ensured for the segment of the flow ($dmax_{D1}$) and share that value with the SDN controller of domain 2. In response, the SDN controller tunes the requirement of delay for the local segment ($dmax_{D2}$) so $dmax_{e2e}$ is ensured. We expect that overprovisioning can be greatly reduced and e2e delay guaranteed by adjusting domain delay budgets dynamically.

IV. DRL-Based Flow Operation

In this section, we first formally describe the DRL engine behind autonomous flow routing. Next, we detail the procedures for the flow provisioning phase and the subsequent near real-time flow operation in single domain scenarios. Finally, extensions to deal with uncertainty of background traffic, as well as for multi-domain operation are presented. Table 1 summarizes the notation used in the rest of the paper.

Table 1 Notation

$x(t)$	input traffic at time t.
$d(t)$	e2e delay measured at time t.
$s(t)$	state at time t.
P	Set of available routes for the flow.
k_p	Capacity of route p.
c_p	Cost of route p.
$y_p(t)$	Traffic routed through route p at time t.
$a_p(t)$	fraction of traffic to be routed through route p at time t.
$r(t)$	reward at time t.
$r_{delay}(t)$	reward component for the obtained delay at time t.
$r_{cost}(t)$	reward component for the routing cost at time t.
α_{delay}, α_{cost}	weights for the rewards in the multi-objective reward function.
β	fixed penalty for violating the maximum delay.
$dmax$	maximum delay to be ensured for the flow.
$xmax$	maximum traffic of the flow.
$xmax$	maximum number of routes for the flow.
$G(V,E)$	graph with the current network state, where V is the set of nodes and E the set of links connecting two nodes.

A. *DRL engine*

From the lessons learned from our previous work in [4], we selected Twin Delayed Deep Deterministic Policy Gradients (TD3) [5], among different DRL techniques. TD3 is an off-policy DRL algorithm that uses a pair of *critic* Deep Neural Networks (DNN) and an *actor* DNN that is updated with some delay. Hereafter, we refer to the set of critic and actor DNNs that run inside the DRL engine as *model*. The model is in charge of dynamically and autonomously deciding which fraction of traffic is routed through each of the allowable routes P that can support the flow.

During near-real-time operation, flow routing agents compute the state, the reward, and the actions periodically (e.g., 1 sec.). Let $x(t)$ be the input traffic and $d(t)$ be the e2e delay, measured at time t. State $s(t)\in\mathbb{R}^+$ is defined as the input traffic scaled by the average capacity of the available routes P, where (k_p) is the capacity of route $p\in P$.

$$s(t) = x(t) \cdot \frac{\sum_{p\in P} k_p}{|P|} \quad (1)$$

Each action $a(t)\in[0,1]^{|P|}$ is a $|P|$-dimensional vector, where every component specifies the fraction of input traffic to be routed through route p. Then, let $y_p(t)$ be the traffic routed through p, i.e.:

$$y_p(t) = a_p(t) \cdot x(t) \quad (2)$$

The reward function $r(t)$ should penalize those actions that resulted into poor QoS or increased network cost. In consequence, a multi-objective reward function with two reward components has been considered to account for the obtained delay (r_{delay}) and for the routing cost (r_{cost}), being α_{delay} and α_{cost} the weight of each component:

$$r(t) = \alpha_{delay} \cdot r_{delay}(t) + \alpha_{cost} \cdot r_{cost}(t) \quad (3)$$

$$r_{delay}(t) = \begin{cases} -\beta - \dfrac{d(t)}{dmax}, & d(t) > dmax \\ 0, & d(t) \le dmax \end{cases} \quad (4)$$

$$r_{cost}(t) = -x(t) \cdot \sum_{p\in P} a_p(t) \cdot \frac{c_p}{k_p} \quad (5)$$

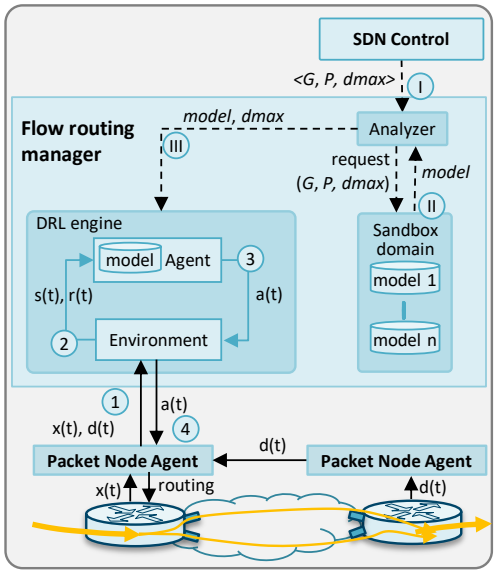

Fig. 5. DRL-based flow operation.

Algorithm 1. Route computation algorithm.

INPUT: $G(V,E)$, req=<s, t, $xmax$, $dmax$, $pmax$>
OUTPUT: P

1: $G_{aux}(V_{aux},E_{aux}) \leftarrow G$
2: **for each** $e \in E_{aux}$ **do**
3: **if** e.cap < $xmax$ **then** E_{aux}.pop(e)
4: $P \leftarrow k\text{-}SP(G_{aux}, s, t)$
5: **for each** $p \in P$ **do**
6: $p.dmin \leftarrow transmDelay(p) + qmin$
7: **if** $p.dmin > dmax$ **then** P.pop(p)
8: sort(P, <'$dmin$','$similarity$'>, ASC)
9: **return** P[1..$pmax$]

Considering that $dmax$ needs to be ensured for the flow, the reward related to the obtained delay can be defined as follows, where β is a fixed penalty for violating the maximum delay. Finally, the reward component for the routing cost is related to the proportion of traffic sent through each of the routes, as well as to the ratio of cost (c_p) and capacity:

B. *Flow provisioning*

Let us start with the very first procedure that is executed at flow provisioning time (Fig. 5). Upon the reception of a new flow provisioning request (req), the SDN controller runs Algorithm 1 to compute the set of allowable routes P. Algorithm 1 receives: *i*) the current network state, summarized in graph $G(V,E)$, where V is the set of nodes and E the set of links, and *ii*) the request including source s and target t nodes, maximum traffic ($xmax$), delay to be guaranteed ($dmax$), and maximum number of allowed routes $pmax$. The algorithm first discards those links with residual capacity below $xmax$ (lines 1-3). Next, the k-shortest path algorithm is used to compute k distinct routes on the resulting graph G_{aux} (lines 4-8). The minimum expected delay $dmin$ (considering both *transmission* and minimum *queuing* and *processing* delay) is computed and used to discard those routes that cannot meet $dmax$. Finally, the remaining routes are sorted by multiple criteria. In this way, the returned set of routes includes those with expected high QoS, while providing high diversity, so agents can choose among

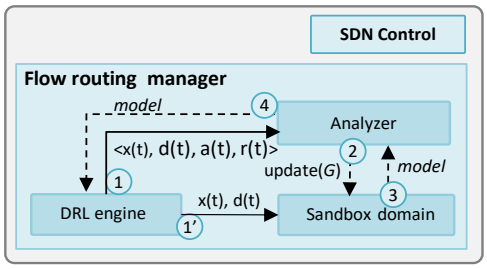

Fig. 6. Flow operation under traffic uncertainty.

alternative options for near-real-time decision making. After route computation, relevant parameters to each $p \in P$, such as the routing cost and capacity are added, which are necessary for autonomous flow routing operation. Note that the complexity of Algorithm 1 is dominated by that of the k-SP algorithm.

Next, the SDN controller initializes the flow routing agents, which contain the following modules (Fig. 5): *i*) the *DRL engine* as described in Section IV.A; *ii*) the *sandbox domain*, that contains pre-trained models and provides support for offline learning; and *iii*) the *analyzer*, in charge of evaluating the performance of models in operation and triggering model updating actions in case of poor performance. Once the flow manager is instantiated, the workflow (in Roman numerals) in Fig. 5 is executed. The topology G, the set of allowable routes P computed by Algorithm 1, and QoS requirement *dmax* are provided to the analyzer (labeled I in Fig. 5). Note that G also contains the maximum traffic volume for the background traffic that is expected for each packet link. Then, the analyzer requests to the sandbox domain an initial model (II). Without loss of generality, we assume that the initial model has been pretrained offline, reproducing a network with the same topology G and routes P as those requested. A generic input traffic, e.g., a sinusoidal daily pattern with some random variation [6], as well as constant background traffic according to configured maxima in G are also assumed. Note that to enable offline training, the specific traffic characteristics are not well known at that time. Therefore, the selection of the generic traffic must verify that pretrained models provide the committed performance when they enter into operation. Finally, the sandbox domain returns the initial model that will be loaded in the DRL engine before operation (III).

C. *Flow operation in single domain scenarios*

Fig. 5 details also the workflow for autonomous flow routing with delay requirements in a single domain scenario (Arabic numerals in Fig. 5). At every time interval t, input traffic $x(t)$ and delay $d(t)$ are collected from the source packet node agent and fed into the DRL environment (1). Note that $x(t)$ can be directly measured at the source packet node, whereas $d(t)$ is computed at the destination and sent to the source node agent. As previously introduced, the environment block is in charge of computing the current state $s(t)$ and reward $r(t)$ and sending them to the DRL agent (2), which is in charge of both learning and decision-making (3). Action vector $a(t)$, with the fraction of traffic to be routed through each of the routes in P, is forwarded to the packet node agent (4). This agent is responsible of translating such proportions into suitable packet node routing configuration and of configuring the routing tables.

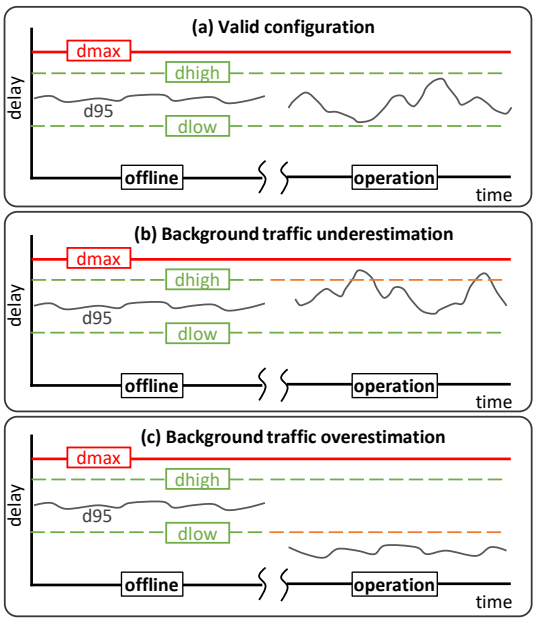

Fig. 7. Detection of background traffic misconfiguration.

D. *Flow operation under traffic uncertainty*

The pre-trained model that is loaded in the DRL engine in the provisioning phase is smoothly improved online from the decisions and actions performed during operation. However, the uncertainty in the input and, more important, the background traffic can lead to poor performance. For instance, if the maximum background traffic is significantly underestimated, large *dmax* violation can be produced, which cannot be corrected by online learning. Then, in order to overcome such issue, Fig. 6 details the proposed workflow that runs in parallel to DRL operation and is intended to improve autonomous operation under traffic uncertainty.

Let us consider that every time the DRL takes and evaluates an action, it pushes tuple $<x(t), d(t), a(t), r(t)>$ to the analyzer (labeled 1 in Fig. 6). At the same time, the monitoring data $x(t)$ and $d(t)$ are also fed to the sandbox domain to tune the offline platform according to real traffic and performance measurements. Then, the analyzer block evaluates the performance of the current model and, if needed, requests the sandbox domain to provide an update of the model with a different configuration for the background traffic (2). The offline trained model that better fits with the new scenario is provided (3) and fed to the DRL engine (4).

The procedure of detecting poor performance due to background traffic misconfiguration (and how to correct it) is illustrated in Fig. 7. The figure shows three different cases of operation that can happen from the same initially offline trained model. Let us assume that the sandbox domain, once the model is trained, can compute the delay range [*dlow*, *dhigh*], for which 95%-percentile of delay (*d95*) is expected to oscillate. Without loss of generality, we consider that *d95* is computed every hour with the last 60 delay values measured each minute. Note that the delay range is computed by simulation as a result of autonomous routing actions and

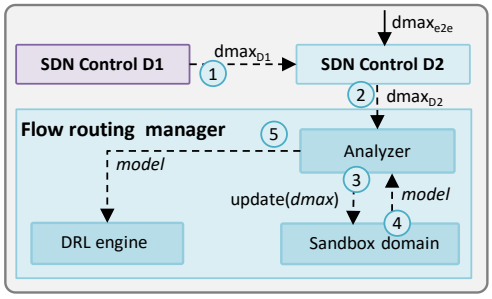

Fig. 8. Flow operation in multi-domain scenarios.

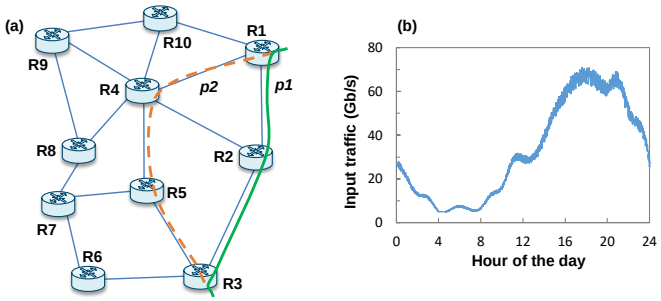

Fig. 9. Network topology (a) and example of input traffic (b).

considering the aforementioned traffic assumptions, i.e., sinusoidal input traffic and constant maximum background traffic. The behavior of *d95* in the sandbox is illustrated in the left part of every figure (labeled *offline*), which stays within delay range, as expected.

Fig. 7a shows a case where the DRL model has a valid performance during operation (right part of the figure). The measured *d95* stays within the range, although it fluctuates more than that assumed during offline training (due to differences between real and simulated traffic and network state). Conversely, Fig. 7b shows a poor performance case induced by underestimation of the real background traffic, which produces large delay, thus violating *dhigh*. Note that, although in the example *dmax* is still guaranteed (not shown in the figure), it is identified as a poor performance case that will trigger model update assuming an increase of background traffic. Alternatively, Fig. 7c shows a case where the *d95* stays below *dlow*, which is considered as an initial overestimation of background traffic. Here, new offline training with a reduction of maximum background traffic will be triggered.

E. Flow operation in multi-domain scenarios

Finally, in order to adapt the proposed autonomous operation to multi-domain scenarios, coordination between domains is implemented to satisfy the e2e delay requirement. Fig. 8 shows the workflow that implements such coordination. For the sake of simplicity, we assume the scenario in Fig. 4, where the flow under study crosses Domain 1 (D1) before entering reference domain D2. Recall that $dmax_{e2e}$ need to be ensured and thus, autonomous operation in D2 needs to guarantee such requirement, considering the delay in the previous domain D1.

Then, once operation starts, the controller of D1 asynchronously notifies its maximum delay $dmax_{D1}$ to the network controller of Domain 2 (labeled 1 in Fig. 8), which computes the requirement for its domain segment as $dmax_{D2} = dmax_{e2e} - dmax_{D1}$ (2). This value is pushed to the flow routing manager, that will work to guarantee such updated $dmax_{D2}$ requirement. In particular, the analyzer asks to the sandbox domain for updating to the new *dmax* (3). The sandbox evaluates whether the current model can properly work with the new QoS requirement; otherwise, return a new model (4) to be loaded into the DRL engine (5). Since the sandbox stores monitored $d(t)$ for a given history, it evaluates whether past $d(t)$ measurements are below new *dmax*. If so, no model update is necessary; otherwise, a new model specifically trained for such new *dmax* is loaded. Note that, in both cases, continuous online learning will improve the model by learning from its routing decisions.

V. Results

In this section, we firstly present the simulation environment used to validate the proposed methodology, models, and algorithms. Then, numerical studies are conducted to validate the proposed models and algorithms in Section IV on: *i*) a simple scenario with no background traffic in single domain networks; and *ii*) challenging scenarios considering known and uncertain background traffic, as well as multi-domain networks.

A. Simulation setup

For evaluation purposes, a Python-based simulator of the control plane containing the modules in Fig. 5 was implemented, which includes: *i*) a lightweight SDN controller; *ii*) the flow routing manager; and *iii*) the packet node agents. The flow routing manager includes a sandbox domain containing a set of models for different *dmax* values trained with a traffic profile following a sinusoidal average daily pattern and random variations following a Gaussian distribution. The data plane was accurately emulated using a simulator based on CURSA-SQ [27] that reproduces realistic traffic flow behavior. CURSA-SQ combines statistically-based traffic flow generation and continuous G/G/1/k queue model based on the logistic function; multiple queuing systems can be numerically analyzed and related statistics can be computed, including queuing delay and packet loss. The accuracy of CURSA-SQ was assessed in [28], where delay and packet loss were compared to those obtained using the well-known NS-3 network simulator in complex scenarios.

Regarding DRL models, a general epsilon-decay strategy was implemented in the TD3 method for balancing between exploration and exploitation, with decay factor equal to 0.00125. Every DNN (actor and critic) consisted of two hidden layers with 100 neurons each implementing the rectified linear unit activation function. All DNNs were trained with the Adam optimizer with learning rate equal to 0.001 in both actor/target actor and critics/target critics, and maximum replay buffer equal to 10^6 samples. A discount factor equal to 0.99 is used to balance between immediate and future rewards. Finally, we set up smoothing factor equal to 0.005 for updating smoothly the actor/critic networks' weight with the target actor/critics networks' weight.

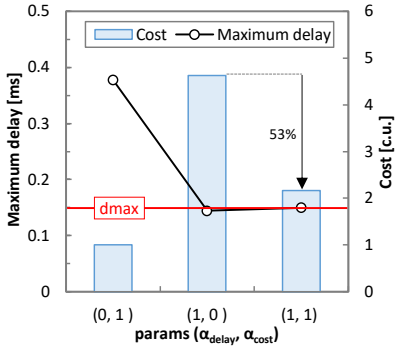

Fig. 10. Overall delay performance under the OPT routing cost scenario.

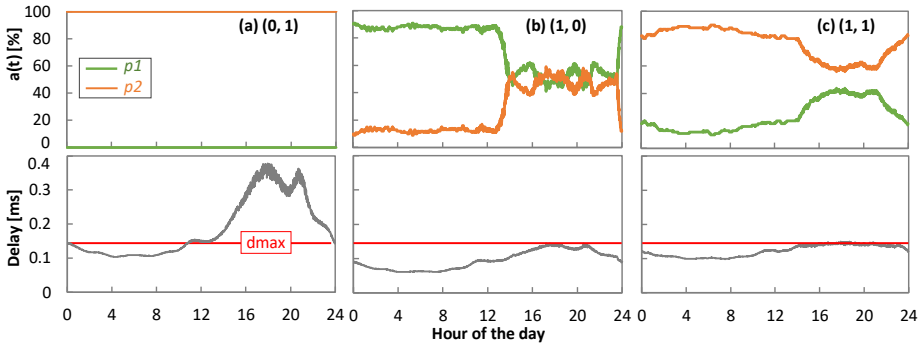

Fig. 11. Detailed performance for configuration (0,1) (a), (1,0) (b), and (1,1) (c) under the OPT routing cost scenario.

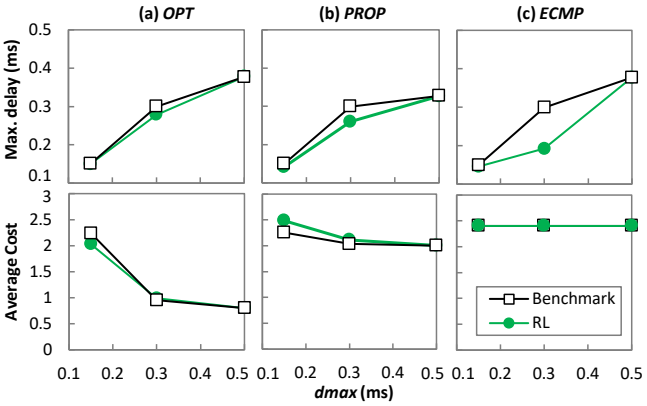

Fig. 12. Delay and cost under different *dmax* and routing cost scenarios.

Table 2 Route cost (c_{pi}) for the selected scenarios

Route	OPT	PROP	ECMP
p1	6	2	3
p2	1	3	3

In single domain scenarios, we consider the Spanish Telefonica's core network topology in Fig. 9a, where all the interfaces are 100Gb/s. Upon the reception of a flow provisioning request (*req*), the SDN controller executes Algorithm 1 and obtains the set of allowable routes *P*. Once the provisioning of the flow routing agent finishes, the flow operation phase starts using the pre-trained offline model that better fits with *req* parameters. Operation was simulated by injecting realistic traffic according to the flow models used in [6]. We assume that the following flow provisioning request: <*s*=R1, *t*=R3, *xmax*=70Gb/s, *dmax*=0.15ms, *pmax*=2>. The following set *P* of allowable routes is computed: {*p1*: <R1, R2, R3>, *p2*:<R1, R4, R5, R3>} (see Fig. 9a), where k_{p1}=k_{p2}=100 Gb/s, being the delay due to transmission and processing 0.05ms and 0.1ms, for *p1* and *p2*, respectively. Fig. 9b shows an example of traffic for the flow during a typical day. For benchmarking purposes, the OSPF routing cost scenarios presented in Table 2 are studied, where: *i*) links weights are computed to the optimality (OPT) as a function of the maximum traffic for each origin-destination pair in the network; *ii*) links weights are inversely proportional to links capacity, as in classical OSPF (PROP); and *iii*) links weights have been changed so to both routes have the same cost (ECMP). Note that the route that provides the best expected QoS performance is *p1* in all the scenarios.

B. *Flow provisioning and operation performance evaluation*

We firstly focus on evaluating the performance of the reward function detailed in eqs. (3)-(5). To this aim, we fix β=12 and compare, under the OPT routing scenario, three different optimization targets by defining different configurations of the tuple (α_{delay}, α_{cost}), namely: (0,1) (*cost minimization*), (1,0) (*delay assurance*), and (1,1) (*multi-objective*). Fig. 10 illustrates the overall performance after 5 days of online operation under all three configurations in terms of maximum delay and average cost. In addition, Fig. 11 shows the detail of the traffic sent thought every route and the delay along a typical day. It is worth highlighting that the DRL engine learned a model producing stable and good-rewarded actions for all the considered configurations.

We observe that the cost minimization configuration (0,1) achieves the best solution in terms of cost, but at the expense of largely violating *dmax*. Fig. 11a shows that traffic is fully routed through the cheapest *p2*. On the opposite side, the delay assurance configuration (1,0) finds a totally different solution, i.e., maximum delay stays below *dmax*, while remarkably increasing routing cost (Fig. 11b). This increment is due to the use of the most expensive *p1*, while reducing the traffic thought *p2*. Interestingly, the multi-objective configuration (1,1) achieves the best overall performance, since maximum delay is below *dmax* and cost is reduced up to 53% to that of configuration (1,0) (Fig. 10). The detailed results in Fig. 11c shows that the DRL engine learned to send more traffic to *p2* without *dmax* violation and consequently, reducing costs. Therefore, we conclude that the proposed DRL-based method is able to converge to different solutions in order to achieve heterogeneous target optimization criteria. In view of the results, we validate the multi-objective configuration (1,1).

Assuming the multi-objective configuration (1,1), let us now evaluate different scenarios for *dmax*, ranging from 0.3 to 1 ms. Fig. 12 shows the maximum delay and average cost for several *dmax* and for the cost scenarios in Table 2. For benchmarking purposes, we plot the performance of a method consisting in assigning *a posteriori* the optimum proportion of traffic $a(t)$ sent to every route that achieves maximum delay as close as possible to *dmax* with the minimum cost. Although this method cannot be applied in real operation (since it

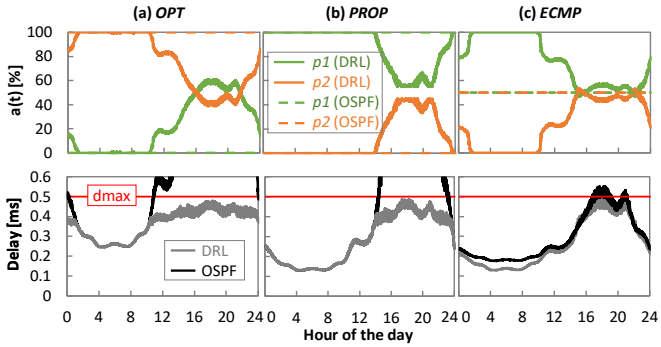

Fig. 13. Comparison of DRL-based flow operation vs. OSPF routing under different routing cost scenarios.

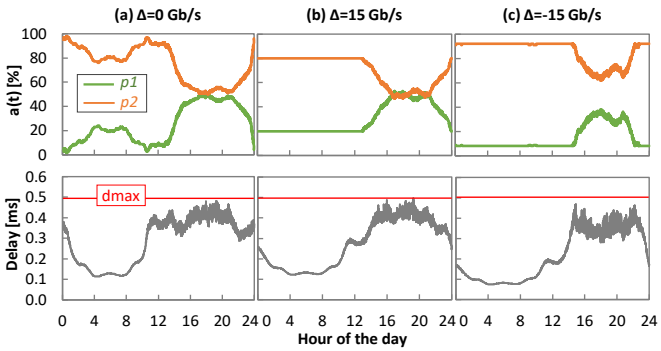

Fig. 14. Performance in the presence of background flows for different Δ configurations under the OPT routing cost scenario.

Table 3 Summary of DRL-based flow operation vs. OSPF routing.

	Max. Delay (ms)		% Time with excessive delay		Avg. Cost	
Scenario	**DRL**	**OSPF**	**DRL**	**OSPF**	**DRL**	**OSPF**
OPT	0.478	11.98	0	55.42	2.93	1
PROP	0.485	11.93	0	36.6	2.23	2
ECMP	0.488	0.551	0	8.89	3	3

assumes perfect knowledge of long-term traffic evolution), it serves as reference for DRL performance evaluation purposes. The results in Fig. 12 show that DRL operation approaches optimum performance for low *dmax* requirements, which anticipates outstanding performance for those B5G/6G services targeting stringent QoS requirements.

C. Evaluation under known background traffic

Let us now evaluate the proposed DRL-based operation for flow *R1-R3* (Fig. 9a) under known background traffic. To that end, on top of the previous traffic configuration, we added two new flows: *R2-R3* with *xmax*=50 Gb/s and *R1-R4* with *xmax*=35 Gb/s. Since the overall network load is higher and delay tends to increase due to larger queued traffic in the interfaces, we relaxed *dmax* from 0.15 to 0.5 ms for flow *R1-R3*. Initial models trained with background traffic of the same *xmax* magnitude were obtained from the sandbox. Moreover, the expected delay variation range for the given background configuration and *dmax* are: [*dlow*, *dhigh*] = [0.375, 0.45]. During flow operation, we assume that the background traffic is constant and equal to *xmax* for both new flows.

We compare the flow operation of the proposed DRL-based approach against using OSPF, under the three routing cost scenarios defined in Table 2. Fig. 13 presents the obtained performance, for the input traffic example in Fig. 9. Both the routing actions $a(t)$ and the measured delay $d(t)$ are shown. We observe that, as in the case with no background traffic shown in Fig. 11c, the DRL agent at the source node balances the traffic between *p1* and *p2* to guarantee *dmax* under all the routing cost scenarios. However, OSPF routing constantly routes packets over the shortest path(s), which translates into flow delay exceeding *dmax* during long periods. Table 3 summarizes the obtained results, where we observe that OSPF exceeds *dmax* during significant periods of time. Interestingly, DRL-based operation, in addition to guarantee *dmax*, is able to minimize average cost. Specifically, cost is

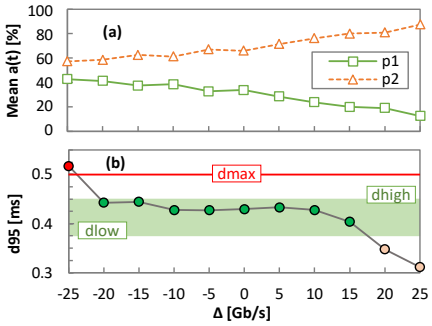

Fig. 15. Routing actions and d95 performance.

close or the same that the one with OSPF for PROP and ECMP scenarios, whereas under the OPT scenario, average cost is higher than with OSPF. The reason behind this performance is that the path that provides the best expected QoS performance, i.e., *p1*, is the most expensive one by far.

D. Evaluation under background traffic uncertainty

The good performance of the previous results could be as a result of having precise knowledge on background traffic volumes. To study that, two alternative scenarios are evaluated, where the difference between actual and expected background traffic (hereafter referred as Δ) is not zero. For this study, we assume traffic flow R1-R4 with *xmax*=20 Gb/s and focus on the OPT routing cost scenario from now on. Fig. 14a shows the performance when *xmax* of the flows are perfectly estimated (Δ=0 Gb/s), Fig. 14b shows the performance when *xmax* of flow R2-R3 is underestimated 15 Gb/s (Δ=-15 Gb/s), while Fig. 14c shows the performance when *xmax* of flow R1-R4 is overestimated 15 Gb/s (Δ=15 Gb/s). In light of the results, we conclude that DRL-based flow operation perfectly adapts to both cases, managing the routing of the flow accordingly to guarantee *dmax* with reduced routing cost.

Fig. 15 summarizes the impact of Δ in terms of mean route usage and *d95*, where range [*dlow*, *dhigh*] is highlighted in green. It can be confirmed that routing tends to use cheap path *p2* when overall network load decreases and *dmax* can be easily accomplished. When Δ>15 Gb/s, *d95* goes below *dlow* and consequently, a new model with lower background traffic should be loaded for operation (although *dmax* is largely guaranteed). On the opposite, when Δ<-20 Gb/s, *d95* exceeds *dhigh*, which will trigger loading a model trained with larger

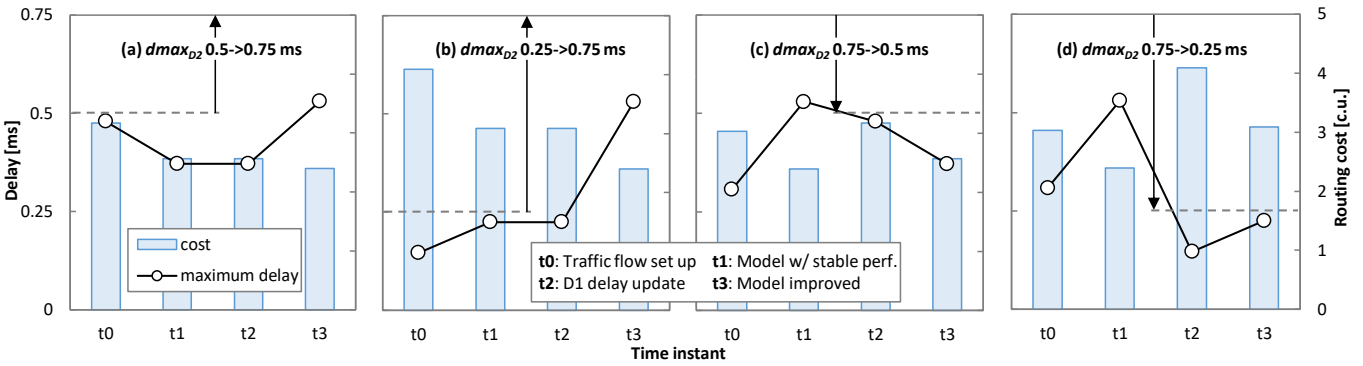

Fig. 16 Performance evaluation in multi-domain scenarios with time-varying background traffic.

background traffic. Therefore, we conclude that the initial model is operative in a large range of background traffic estimation error. This is a very relevant result because it shows that only a prior approximate (not very accurate) knowledge of background traffic is enough to guarantee high performance of the DRL-based operation.

E. Evaluation in multi-domain scenarios

Last but not least, we focus on multi-domain scenarios and consider the scenario in Fig. 4. Specifically, we focus on the actions carried out in domain D2, which topology is assumed to be that of Fig. 9a; now, flow R1-R3 is the segment in D2 of e2e flow R1.A-R2.Z, where R2.B in Fig. 4 is R1 and R2.Z is R3. Let us consider $dmax_{e2e}$=1ms.

As in the previous subsection, two background traffic flows R2-R3 and R1-R4 with *xmax* equal to 50 Gb/s and 20 Gb/s, respectively, are established in domain D2. However, to make the scenario more realistic and challenging, in this subsection, we assume that background traffic is not constant in time, but it fluctuates following a time-varying sinusoidal daily pattern. Therefore, the objective of this performance evaluation is two-fold: *i*) to evaluate the workflow for multi-domain scenarios proposed in Section IV.E; and *ii*) to analyze the impact of time-varying background traffic flows.

Aiming at evaluating the aforementioned objective, we conducted simulations in the following way. First, and similarly to previous sections, the new flow segment R1-R3 was set up in D2 at time t0 and a pre-trained initial model assuming constant background traffic and a given $dmax_{D2}$ was used for operation. This model is continuously improved through online learning; note that it now needs to learn the actual characteristics of the input traffic and those of the time-varying background traffic. After some time in operation, at time *t1* the model reaches a stable performance that cannot be significantly further improved. Then, at time t2, the D2 SDN controller receives an asynchronous notification from D1 SDN controller updating $dmax_{D1}$, which in turn triggers updating $dmax_{D2}$ and consequently, the proposed analysis and model update procedure is carried out. Then, operation continues with the new $dmax_{D2}$. Because of the changes, online learning might improve the model in operation, which will reach performance stability at time *t3*.

Fig. 16 summarizes the main results of the simulations in terms of routing cost and delay measured at every of the abovementioned time instants. Specifically, Fig. 16a and Fig. 16b show two cases, where $dmax_{D2}$ is relaxed (from 0.5 to 0.75ms and from 0.25 to 0.75ms, respectively), whereas in Fig. 16c and Fig. 16d $dmax_{D2}$ becomes more stringent (from 0.75 to 0.5 and from 0.75 to 0.25, respectively).

We observe that online learning improves initial pre-trained models even in the presence of time-varying background traffic, since routing cost is reduced in all the cases from t0 to t1, while $dmax_{D2}$ is guaranteed in the whole period [t0, t1]. In case of Fig. 16a and Fig. 16b, there was no change in the model in time t2 because of $dmax_{D2}$ relaxation and hence, performance in t2 equals that of t1. However, a new model was loaded when $dmax_{D2}$ was reduced in time t2, and which reduced maximum delay to guarantee the desired QoS performance from t2 on, as observed in Fig. 16c and Fig. 16d. Finally, note that regardless the case, the model was improved after $dmax_{D2}$ update, by increasing maximum delay and/or reducing routing cost.

In view of these results, we can definitely validate the proposed methodology for DRL-based operation in the presence of time-varying background traffic flows and in multi-domain scenarios.

VI. Concluding Remarks

Distributed near real-time autonomous flow routing with QoS assurance has been proposed. The solution is based on multiple agents that are able to make decisions in a collaborative way. At provisioning time, the SDN controller provides the agents with a set or paths that can be used for routing the traffic flow, so the agents can make routing decisions using only those paths. Agents include a robust DRL engine to learn from the flow routing decisions with the objective to minimize routing costs while ensuring the committed QoS in terms of e2e delay. DRL management lifecycle was considered, where models are continuously improved for taking robust actions from the beginning of flow operation.

The performance of the proposed DRL-based solution has been shown under several challenging scenarios. First of all, three different optimization targets were studied, i.e., cost minimization, delay assurance, and multi-objective, under the assumption of constant and known background traffic. The results showed that the multi-objective option was able to assure e2e delay and minimize flow routing costs, reducing them by 53% with respect to just considering delay assurance.

Results illustrated how the different options made near real-time routing decisions on which proportion of traffic should be routing through each available path. Different delay configurations and cost scenarios were also studied, showing that the proposed solution achieves performance closed to the optimal one.

For benchmarking purposes, the proposed DRL-based routing was compared against OSPF for three cost scenarios. The results showed DRL-based flow operation achieves target QoS with minimum cost in contrast to OSPF, which is unable to guarantee reliable and consistent flow QoS performance.

Next, evaluation of the solution under background traffic uncertainty was carried out, where that traffic was constant and variable with time. The impact of background traffic estimation was analyzed and concluded that the DRL model in operation provided good performance on a large range of background traffic volumes over a nominal one, e.g., [-20, 15] Gb/s. This fact validates the adopted DRL management lifecycle, since it enables pre-training different DRL models in a sandbox domain for a small set of nominal background traffic volumes, while reducing the amount of model changes.

Finally, the performance of our solution was evaluated in multi-domain scenarios, where the delay to be ensured in a domain can change as a result of the traffic conditions and the operation in other domains. In this case, model selection demonstrated to adapt agents' operation in the event of a change in the delay to be ensured in the current domain. In addition, online learning was shown to improve even further the performance of the model in operation.

REFERENCES

[1] *Beyond 5G/6G KPIs and Target Values* [white paper], 5GPPP, v1, 2022.

[2] D. Rafique and L. Velasco, "Machine Learning for Optical Network Automation: Overview, Architecture and Applications," J. of Optical Communications and Networking, vol. 10, pp. D126-D143, 2018.

[3] L. Velasco, A. Chiadò Piat, O. González, A. Lord, A. Napoli, P. Layec, D. Rafique, A. D'Errico, D. King, M. Ruiz, F. Cugini, and R. Casellas, "Monitoring and Data Analytics for Optical Networking: Benefits, Architectures, and Use Cases," Network Magazine, vol. 33, pp. 100-108, 2019.

[4] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," MIT Press, Cambridge, UK, 2nd ed., 2020.

[5] V. Francois-Lavet, P. Henderson, R. Islam, M. Bellemare, J. Pineau, "An Introduction to Deep Reinforcement Learning," Foundations and Trends in ML, vol. 11, pp. 219-354, 2018.

[6] S. Barzegar, M. Ruiz, and L. Velasco, "Packet Flow Capacity Autonomous Operation based on Reinforcement Learning," MDPI Sensors, vol. 21, pp. 8306, 2021.

[7] L. Velasco, S. Barzegar, D. Sequeira, A. Ferrari, N. Costa, V. Curri, J. Pedro, A. Napoli, and M. Ruiz, "Autonomous and Energy Efficient Lightpath Operation based on Digital Subcarrier Multiplexing," J. on Selected Areas in Communications, vol. 39, pp. 2864-2877, 2021.

[8] S. Barzegar, M. Ruiz, and L. Velasco, "Distributed and Autonomous Flow Routing Based on Deep Reinforcement Learning," in Proc. Int. Conference on Photonics in Switching and Computing (PSC), 2022.

[9] M. Wooldridge, *An introduction to multiagent systems*, John Wiley & Sons, 2009.

[10] J. Moy, "OSPF version 2," The Internet Engineering Task Force, Request for Comments 2328, 1998.

[11] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in Proc. IEEE INFOCOM, vol. 2, pp. 519–528, 2000.

[12] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to Route," in Proc. of ACM Workshop on Hot Topics in Networks, 2017.

[13] P. Almasan, J. Suarez-Varela, B. Wu, S. Xiao, P. Barlet-Ros and A. Cabellos-Aparicio, "Towards Real-Time Routing Optimization with Deep Reinforcement Learning: Open Challenges," in Proc. Int. Conference on High Performance Switching and Routing, 2021.

[14] G. Bernardez J. Suarez-Varela, A. Lopez, X. Shi, S. Xiao, X. Cheng, P. Barlet, and A. Cabellos, "Multi-Agent System for Traffic Engineering," IEEE Transactions on Cognitive Communications and Networking, vol. 9, 2023.

[15] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, J. Arco, "A survey on machine learning techniques for routing optimization in SDN," IEEE Access, vol. 9, pp. 104582-104611, 2021.

[16] O. Pedrola, A. Castro, L. Velasco, M. Ruiz, J. Fernández-Palacios, D. Careglio, "CAPEX study for Multilayer IP/MPLS over Flexgrid Optical Network," J of Optical Communications and Networking, vol. 4, pp. 639-650, 2012.

[17] Y. Chen, A. Rezapour, W. Tzeng and S. Tsai, "RL-Routing: An SDN Routing Algorithm Based on Deep Reinforcement Learning," Transactions on Network Science and Engineering, vol. 7, pp. 3185-3199, 2020.

[18] O. Fares, A. Dandoush, and N. Aitsaadi, "OPR: SDN-based Optimal Path Routing within Transit Autonomous System Networks," in proc. IEEE International Conference on Communications, pp. 1058-1063, 2022.

[19] S. Lin, I. Akyildiz, P. Wang and M. Luo, "QoS-Aware Adaptive Routing in Multi-layer Hierarchical Software Defined Networks: A Reinforcement Learning Approach," in proc. of Int. Conf. on Services Computing (SCC), 2016.

[20] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, C. Paasch (Eds), "TCP Extensions for Multipath Operation with Multiple Addresses," IETF RFC 8684, 2020.

[21] S. Lee and J. Yoo, "Reinforcement Learning Based Multipath QUIC Scheduler for Multimedia Streaming," Sensors, vol. 22, pp. 6333, 2022.

[22] A. Mellouk and S. Hoceini, "Reinforcing State-Dependent N Best Quality of Service Routes in Communication Networks," in Proc. Workshop on High Performance Switching and Routing, 2007.

[23] H. An, Y. Ji, N. Zhang, W. Hu, P. Yu and Y. Wang, "Dynamically Split the Traffic in Software Defined Network Based on Deep Reinforcement Learning," in proc. Int. Wireless Communications and Mobile Computing, 2020.

[24] J. Rischke, P. Sossalla, H. Salah, F. Fitzek and M. Reisslein, "QR-SDN: Towards Reinforcement Learning States, Actions, and Rewards for Direct Flow Routing in Software-Defined Networks," IEEE Access, vol. 8, pp. 174773-174791, 2020.

[25] J. Zhang, M. Ye, Z. Guo, C. Yen and H. Chao, "CFR-RL: Traffic Engineering with Reinforcement Learning in SDN," in J. on Sel. Areas in Communications, vol. 38, pp. 2249-2259, 2020.

[26] S. Barzegar, M. Ruiz, A. Sgambelluri, F. Cugini, A. Napoli, and L. Velasco, "Soft-Failure Detection, Localization, Identification,

and Severity Prediction by Estimating QoT Model Input Parameters," Transactions on Network and Service Management, vol. 18, pp. 2627-2640, 2021.

[27] M. Ruiz, F. Coltraro, and L. Velasco, "CURSA-SQ: A Methodology for Service-Centric Traffic Flow Analysis," J of Optical Communications and Networking, vol. 10, pp. 773-784, 2018.

[28] A. Bernal, M. Richart, M. Ruiz, A. Castro, and L. Velasco, "Near Real-Time Estimation of End-to-End Performance in Converged Fixed-Mobile Networks," Elsevier Computer Communications, vol. 150, pp. 393-404, 2020.