Autonomous Flow Routing Based on Deep Reinforcement Learning

S. Barzegar, H. Shakespear-Miles, M. Ruiz, and L. Velasco
Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
e-mail: hailey.shakespear@upc.edu

ABSTRACT

The progression of 5G and beyond brings forth a demand for dynamic services and with it many service requirements, prompting the need for adaptable and automated solutions. Existing autonomous network operations, reliant on Machine Learning, often grapple with real-time responsiveness due to their dependency on a global network perspective. In this work, we show a distributed approach tailored for assuring QoS autonomously with near-real-time flow routing. The strategy centers on reducing response times by decentralizing intelligence nearer to the data plane. Utilizing Deep Reinforcement Learning (DRL) allows for the solution to be suitable in dynamic scenarios. Through distributed decision-making, node agents ensure QoS, specifically focusing on end-to-end delay, while simultaneously optimizing routing costs. Results underscore the effectiveness of our approach in dynamic scenarios, showcasing its applicability without prior knowledge of traffic profiles.

Keywords: ***Distributed Control, Autonomous systems, Deep Reinforcement Learning***

1. INTRODUCTION

The expected traffic trends for beyond 5G (B5G) and 6G services include large traffic dynamicity and increased demands, meaning that transport networks must be redesigned to support these demands while meeting stringent performance requirements. To meet such goals, networks must be adapted to include higher levels of flexibility and leverage autonomation. In this way, other considerations like network optimization and cost efficiency can be given a higher priority during operation. In recent years many solutions based on Artificial Intelligence (AI) and Machine Learning (ML) have been proposed as a way to introduce autonomous network operation [1]. These solutions allow for the implementation of closed control loops based on data. These solutions typically run in some centralized element where they have a global view of the network, potentially reducing the cost of operation by minimizing human intervention. Due to this, current network architectures rely on Software Defined Networking (SDN) controllers as they can be used to carry out the decision making in a centralized manner.

However, when making (near) real-time decisions this centralized location can be a disadvantage, especially when considering highly dynamic traffic conditions. The long response times associated with centralized decision making can lead to poor resource utilization as the decision made may no longer be relevant. This work focuses on flow routing where routing decisions are made near real time. In this way resource utilization can be optimized while ensuring the Quality of Service (QoS) of the flows. In this case, traffic variations can create bottlenecks that impact the time required for transmitting the flow traffic between two boarder packet nodes, referred to as end-to-end (e2e) delay. If the response time for routing decision making is large, it cannot respond quickly enough to these traffic variations and the e2e delay can be affected. One approach to reducing response times is to execute the decision making as closely to the data sources as possible. This not only reduces response times but also the amount of data necessary for such decisions. To this end, a possible solution is to use Deep Reinforcement Leaning (DRL) [2], [3] as it requires few data points in training and is highly suitable for real-time decision making [4].

This work summarizes the approach in [5], for distributed autonomous flow routing based on DRL algorithm running in the packet nodes. In the scope of networking, agents are deployed at the nodes to make autonomous decisions in near-real-time based on specific network requirements received from the SDN controller. The SDN controller is then not required for near-real-time operations. The DRL algorithms are tasked with the near-real-time decision making routing the packet flows taking into consideration the measured e2e delay. This type of algorithm is especially suited to the task as no previous knowledge of the traffic is necessary as it is able to learn relevant traffic characteristics.

2. FLOW ROUTING DISTRIBUTED FRAMEWORK

In traditional approaches for flow routing that use SDN controllers, the SDN controller itself computes the routes for the packets in the flow to follow. This can be performed at flow provisioning time based on the information available to the SDN controller like network topology and current network conditions.

However, when distributing the intelligence, the role of the SDN controller becomes reduced. Figure 1 outlines the general distributed approach and highlights the new roles of each component. For simplicity only a single node is shown in detail along with the SDN controller. From the figure we can see that the intelligence is distributed to each router introducing the need for some agents. A multi-agent system (MAS) framework can then be used for coordination and communication among the different agents and the SDN controller. With the intelligence distributed down, we obtain some hybrid centralized SDN control where the SDN controller is responsible for the generation coordination of the network that takes advantage of its global view providing guidelines to the agents.

979-8-3503-7732-3/24/$31.00

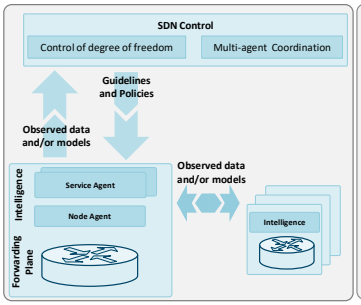

Figure 1: Distributed intelligence framework.

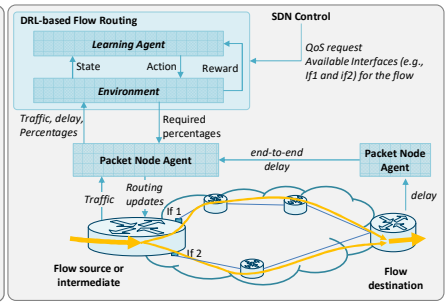

Figure 2: Example scenario of distributed flow routing.

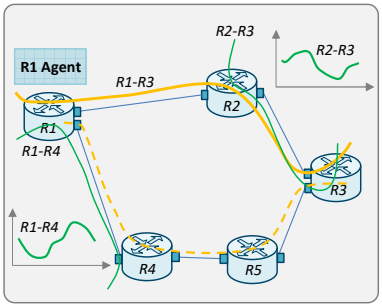

Figure 3: Operation under uncertain background traffic.

The decision making is then performed by each agent locally, based on the observed data and another other information shared between agents.

Figure 2 depicts the process for intelligent flow routing decisions at a packet node. The SDN controller provides QoS requirements, the two routes available, and associated costs. The DRL agent consists of a learning agent and an environment which it uses to optimize route selection to meet the committed QoS ($dmax_{e2e}$), while minimizing the cost of using each route. The system learns optimal actions based on the network conditions like the traffic, delay and current routing percentages, which it uses to calculate the current state and rewards received. This state is used by the learning agent who decides the appropriate action and returns it back to the packet node agent as the required percentages which are implemented. Traffic is routed through two interfaces, with multiple sub-flows following the same route. Upon reaching the destination, e2e delay is measured, statistics computed, and relayed to participating node agents. The process is then repeated for the rest of operation.

If the traffic varies in a predictable way, such as typical daily variation, the evolution can be predicted fairly simply. However, intelligence is needed when considering variation in the traffic and network dynamicity. Figure 3, depicts a more complex network scenario with 5 packet nodes and 3 packet flows shown as R2-R3, R1-R4 (green) and R1-R3 (yellow solid line). 2 of these packet flows (green) vary in time. The R1 Agent receives two possible routes from the SDN controller (yellow). First, all traffic is routed via the shortest path, R1-R2, ensuring acceptable delay for flow R1-R3. However, when flow R2-R3 is established, the agent must divert the traffic to the other longer and more costly route R1-R4-R5-R3, to mitigate the delay and make sure it doesn't exceed the maximum allowed value. This adaptive routing continues as traffic and delay conditions fluctuate, with R1 adjusting routes to optimize either delay or cost.

3. DRL ENGINE

This section begins by explaining how the DRL models are integrated and then goes into detail on the DRL design. Firstly, in Figure 4, the SDN controller initializes the necessary agents including the DRL engine; the sandbox domain with pre-trained models; the analyzer which evaluates model performance. The workflow in **Error! Reference source not found.** proceeds as follows: I) Analyzer receives topology *G*, available routes *P*, and QoS requirement *dmax*. With this information it can request the initial model to be used in operation from the sandbox domain in II). The Sandbox domain returns the pre-trained model and it is loaded into the DRL engine in III). This is the model that the DRL engine will use for decision making until another model is passed to it by the Analyzer. The decision-making process is executed as follows.

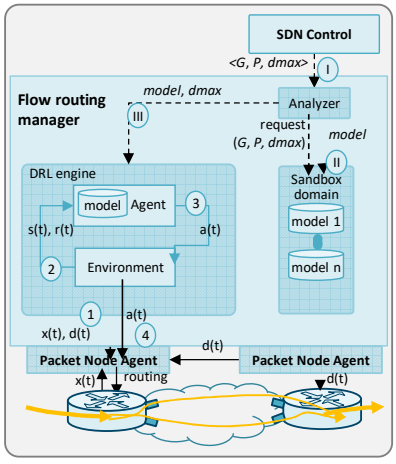

Figure 4: DRL-based flow routing in operation.

For every time interval t, measurements such as the input traffic *x(t)* and the delay *d(t)* are obtained from the source packet node agent (1). The delay is calculated at the destination packet node and sent to the source packed node which communicates it with the DRL engine. As mentioned previously, the environment calculates the state, *s(t)*, and reward, *r(t),* based on the collected information from the packet node agent. It shares this information with the DRL Agent (2) where the learning and decision making takes place. It decides the most appropriate action *a(t)* which describes the percentage of traffic to be sent through each available routes in P (3). This sent to the packet node agent (4) to translate the request into corresponding packet routing configurations and implementing them.

There are many DRL algorithms that can be used for agent learning. In this case Twin Delayed Deep Deterministic Policy Gradients (TD3) was selected to for intelligent decision making. It uses a pair of *critic* Deep

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

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

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

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

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

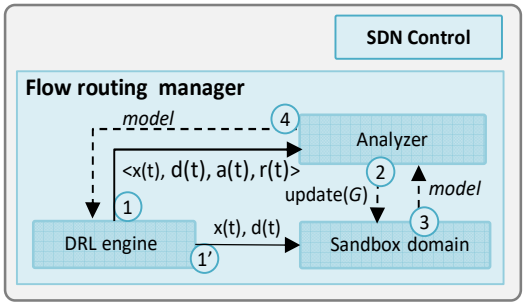

Figure 5: Flow routing manager operation.

Neural Networks (DNN) and an *actor* DNN that is updated with some delay [3]. For simplicity, the set of DNNs is referred to as the DRL model and is the model that will be used to perform the routing decisions for a given source packet node agent.

Following Figure 4, the state, action and reward can be further defined. Firstly, the state is defined in Eq. (1) as the input traffic scaled by the average capacity available in all the routed. The capacity can be defined as k_p which describes the capacity available in a given path. Secondly, the action can be defined as a vector of length P, where each element is between 0 and 1 and describes the fraction of input traffic that is to be sent through the corresponding route. This can then be used to calculate an additional metric $y_p(t)$ shown in Eq. 2 which describes the amount of traffic routed through a given path. That is to say the corresponding action for that path times the total input traffic gives the traffic sent through that given path. Finally, the reward function can be defined in Eq. 3. The reward function takes into account two components weighted by α_{delay} and α_{cost}. This makes it a multi-objective reward function where it seeks both to minimize the e2e delay as well as the cost of using the routes produced by the actions. The first component is the delay component which is defined in Eq. 4. If the e2e delay returned is under the *dmax* there is no reward penalization. However, if the *dmax* is violated the reward returns the violation with some additional penalty β. Likewise, the cost component of the reward function is shown in Eq. 5. Here a total cost is calculated by taking the ratio of route cost cp to the route capacity and multiplying it by the proportion of traffic sent to each route. This means that the cost is directly related to the path in use and how much traffic is being sent down that path.

The relationship between the DRL engine, analyzer and sandbox domain introduced in Figure 4 can be elaborated on further. Figure 5 illustrates the relationship between the three parts and their interactions. For each decision made by the DRL engine it provides the analyzer with a tuple $<x(t), d(t), a(t), r(t)>$ containing the input traffic, the e2e delay, the chosen actions, and the reward obtained (1). Simultaneously it provides the sandbox domain with the input traffic and the obtained e2e delay (1'). In this way the models being trained in the sandbox have the up-to-date performance measurements that it can use for offline tuning of models. The analyzer receives the tuple from the DRL engine and evaluates the current performance. If it determines that the model is not performing satisfactorily it requests a new model from the sandbox domain (2). The poor model performance can be due to many factors, for example, variations or changes in the input and background traffic that are underestimated by the model, or changes in the topology G that come from the SDN controller. In these cases, the sandbox domain returns the model that better suits the current scenario to analyzer (3), who in turn provides it to the DRL engine for operation (4). Once the DRL engine receives the model it can being operating with it and slowly improve it through online learning using the state, actions, and rewards explained previously. In general, it can be said that the online learning performed by the DRL engine is for fine tuning, where large model changes are supervised by the analyzer and performed by the sandbox domain, the combination of both providing the best model.

5. RESULTS

Testing was conducted in a python-based simulator where realistic traffic flow behavior was used for a similar topology to the one shown in Figure 3. Before considering model performance for e2e delay ensure, the multi-objective reward function can be evaluated. By fixing the delay penalization β=12 we can consider different combinations of (α_{delay}, α_{cost}) to evaluate the effectiveness of the multi-objective reward function. To this end three different cases can be studied: (0,1) where only the cost of the routes is considered, (1,0) where only the delay is considered, and (1,1) the full multi-objective function. Figure 6 shows the performance of each of these reward function cases showing the maximum delay and cost obtained after 5 days of operation. It is clear that only considering the cost is not enough to ensure the *dmax* and the multi-objective function is able to significantly reduce the cost compared to only considering e2e delay. This can be examined in more detail in Figure 7 showing both the percent of traffic sent to each path and the *dmax* obtained. Here the advantage of the multi-objective reward can be truly shown, as in Figure 7 (a) by fully routing to path 2 (p2) the cost is kept a minimum, where delay suffers greatly. Conversely Figure 7 (b) relies on path 1 (p1) to obtain a low *dmax* but produces a high cost. Then in Figure 7 (c) a slightly higher dmax is obtained but the cost significantly reduces as it relies mostly on p1

979-8-3503-7732-3/24/$31.00 ©2024 IEEE

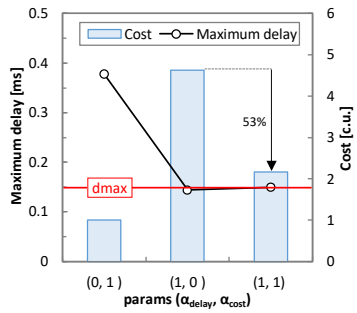

Figure 6: Overall delay performance under different reward functions.

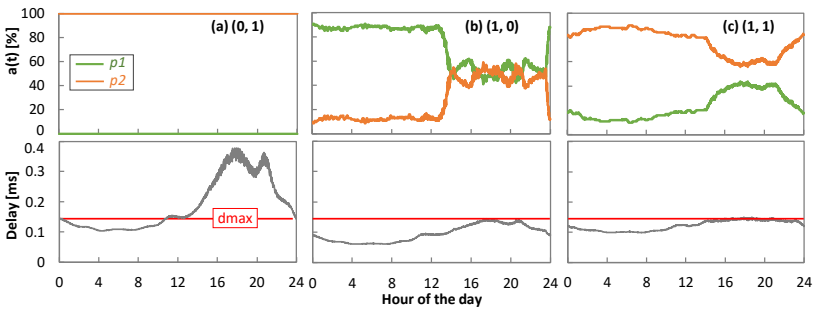

Figure 7: Detailed performance for configuration (0,1) (a), (1,0) (b), and (1,1) (c)

for routing. The multi-objective function has been validated and is demonstrable the most suitable. The rest of the results are then obtained using the multi-objective reward function.

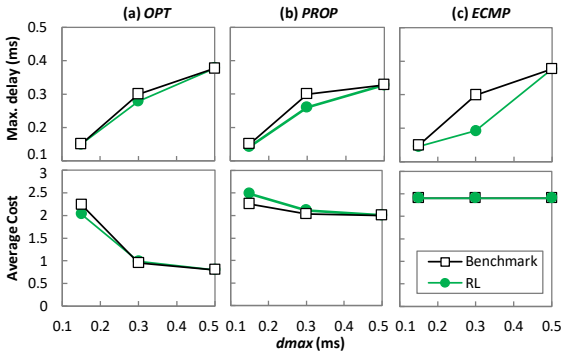

Figure 8: Delay and cost under different *dmax* and routing cost scenarios.

Next, the performance of the DRL engines can be examined by studying various *dmax* requirements for different cost scenarios. The *dmax* values considered are 0.15, 0.3 and 0.5. While the different cost scenarios considered are i) link weights are optimally calculated based on the maximum traffic for each origin-destination pair in the network (OPT); ii) link weights are inversely related to link capacity (PROP); and iii) link weights have been adjusted to ensure that both routes have equal costs (ECMP). A benchmark is considered as reference where percentages are calculated a posteriori to show what the optimal allocation would be if there was perfect knowledge of network and traffic conditions.

Figure 8 shows the maximum delay (upper) and average cost (lower) for the different scenarios proposed. In the OPT and PROP cases, Figure 8 (a) and (b) respectively, the model achieves optimal or almost optimal performance for both delay and average cost values. In Figure 8 (c), the cost is optimal while there are some differences in the maximum delay of the ECMP scenario. However, at low *dmax* values the behavior is optimal in the ECMP as well as the other cases, which critical when considering stringent QoS requirements.

6. CONCLUSION

The approach for autonomous flow routing using a hybrid centralized-decentralized control architecture was discussed in this work. It combines the global oversight of SDN with the local responsiveness of DRL agents by focusing on the implementation of distributed autonomous flow routing based on DRL algorithms within a MAS framework. This facilitates near-real-time decision-making directly at packet nodes. Particularly by using TD3 models, within this framework, the decision-making was addressed through both e2e delay and network costs. Simulation results validate the superiority of the multi-objective reward function in achieving optimal network performance, demonstrating that DRL-based solutions are well-suited for managing the complexities of future network environments when applied to distributed autonomous flow routing.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon Europe research and innovation programme DESIRE6G (G.A. 101096466) and from the MICINN IBON (PID2020-114135RB-I00) project and from the ICREA Institution.

REFERENCES

[1] 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.

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

[3] 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.

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

[5] S. Barzegar, M. Ruiz and L. Velasco, "Autonomous Flow Routing for Near Real-Time Quality of Service Assurance," IEEE Transactions on Network and Service Management (TNSM) *2023*

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