Optimizing Time-Sensitive Applications in Distributed SDNs with Reinforcement Learning

Ioannis Panitsas, Akrit Mudvari, Leandros Tassiulas
Department of Electrical Engineering, Yale University, New Haven, CT, USA
Emails: {ioannis.panitsas, akrit.mudvari, leandros.tassiulas}@yale.edu

***Abstract*—In distributed Software-Defined Networking (SDN), multiple physical SDN controllers are deployed to achieve centralized view and control of the entire network by synchronizing with each other. Despite the availability of various synchronization policies for distributed SDN controller architectures, most current research works do not consider joint optimization of network and user performance. In this paper, we explore learning-based synchronization policies designed to minimize network operational costs by strategically offloading time-sensitive tasks to proximate edge servers while ensuring that the latency requirements are met. We formulate the controller synchronization problem as a Markov Decision Process (MDP) and employ Reinforcement Learning (RL) techniques to enhance both user satisfaction and network profitability. Evaluation results in various simulated SDN environments demonstrate that value-based RL approaches increase the total number of latency-compliant paths by 15.18% while simultaneously reducing operation costs by 22.95%, compared to policy-based RL approaches and heuristics.**

***Index Terms*—Software-Defined Networking (SDN), Reinforcement Learning (RL)**

I. Introduction

Software-Defined Networking (SDN) introduces a groundbreaking shift in network architecture, fundamentally differentiating the decision-making entity, known as the control plane, from the physical elements that handle user data, referred to as the data plane [1]. SDN applications can be implemented and deployed in the application plane, where northbound interfaces connect the control with the application plane, and southbound interfaces connect the control with the data plane. This separation enhances network performance through programmable management and reconfiguration, offering network operators increased flexibility and adaptability. In this centralized architecture, all the network control functionalities are implemented in an SDN controller, which possesses the entire network state. While centralized approaches offer significant benefits, they encounter challenges related to scalability, availability, and security. As network infrastructures become more complex, centralized controllers face limitations in computational capacity and lack scalability. Additionally, concentrating control plane functions in a single node introduces security vulnerabilities, potentially compromising network resilience and reliability. To overcome these issues, a distributed SDN approach, with multiple hierarchical and independent controllers managing specific network domains, has been proposed in the literature [2]. These controllers, though physically separated, operate in a 'logically-centralized' manner by periodically exchanging domain network states for global decision-making, a procedure known as *controller synchronization*. In large-scale networks, complete synchronization among controllers is often cost-prohibitive due to high communication demands, leading many distributed SDN networks to adopt partial synchronization and accept temporary inconsistencies in controllers' network views, which is known as the *eventual consistency model* [3].

Utilizing this model, several research works have been proposed to optimize various SDN applications by approximating the synchronization policy using Reinforcement Learning (RL) techniques. In [4], the authors presented a systematic methodology for deciding the optimal frequency of exchanging synchronization messages among controllers for shortest path routing and load balancing SDN applications. The authors in [5], proposed a Deep Reinforcement Learning (DRL) algorithm to minimize the communication latency and waiting time of inter-domain routing tasks by selectively synchronizing the optimal subset of SDN controllers. In this work [6], the authors introduced a DRL-based synchronization policy for inter-domain routing tasks. In [7], DRL and transfer learning synchronization policies were developed for inter-domain routing and load balancing, respectively. In this study [8], a joint controller synchronization and placement RL algorithm was proposed and tested in inter-domain routing and load balancing SDN applications.

Despite extensive efforts to optimize individual network metrics, there is a conspicuous lack of proposals addressing controller synchronization designs specifically tailored to time-sensitive applications, such as Augmented Reality (AR) and Virtual Reality (VR) technologies, which demand low latency and high computational resources to ensure smooth and immersive experiences to the end users. Such designs should jointly optimize SDN application performance and network efficiency, providing simultaneous benefits to both users and network operators. To fill this gap, we have developed an SDN application called "AR/VR Optimization" to minimize network operator costs by strategically offloading user tasks to the most cost-effective servers while ensuring that the predefined latency requirements are met. We formulate this optimization problem as a Markov Decision Process (MDP) and utilize RL techniques to approximate the optimal synchronization policy. Evaluation results demonstrate that value-based approaches outperform both policy-based methods and heuristics, increasing the number of latency-compliant paths by 15.18% and reducing network operator costs by 22.95%.

II. System Description

A. Distributed SDN Environment

The network architecture comprises distributed SDN controllers (C) and data plane devices (D). Each network domain includes SDN-enabled data plane devices, such as programmable switches and routers, alongside edge servers for task offloading and real-time AR/VR application processing. These domains are interconnected through gateway nodes with inter-domain links. Distributed SDN controllers manage and control their own domain, maintaining a comprehensive and up-to-date view by continuously monitoring intra- and inter-domain link and node-level statistics, including link latency, throughput, and computing resource utilization. The network domains are highly heterogeneous and dynamic, as each can contain a different number of routing and computation devices to cover the varying user traffic demands. The available capacity of intra- and inter-domain links is constantly evolving due to the background traffic in each domain, as well as the available computing resources in edge servers. Finally, we assume that each edge server is assigned a dynamic cost variable, representing the processing cost for completing a task. This cost variable can alternatively be interpreted as the amount of energy the server will spend to fulfill the computing requirements of the task. In this work, we assume that the queuing waiting time in the SDN-enabled switches and the server processing time for each task are constant across all switches and servers respectively.

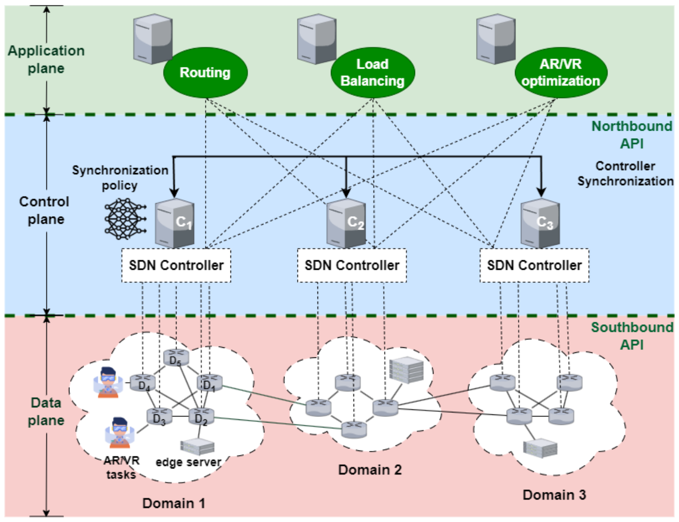

Fig. 1. A network environment with multiple distributed SDN controllers. The synchronization policy is deployed in C_1 controller, and guides C_1, which controllers to synchronize at each period to maintain a global network view, for optimizing the network performance metric.

B. Application of Interest

In this work, we focus on an SDN application which leverages both intra- and inter-domain network state information. Due to limited synchronization, each controller possesses only a partial view of the entire network. Considering this constraint, the objective of this SDN application is to strategically offload user tasks to the most cost-effective servers while ensuring that latency constraints are satisfied. This SDN application is particularly suitable for latency-sensitive applications such as AR/VR, due to the stringent latency requirements, and it also offers significant benefits for network operators aiming to minimize operational costs. Henceforth, we will refer to this application as the ”AR/VR Optimization” SDN application.

C. Performance Metric

User tasks associated with AR/VR applications, characterized by diverse latency requirements, are aggregated within each domain and subsequently forwarded to the respective domain-specific SDN controller. These tasks are then processed and queued by the SDN controller. Leveraging its network view, the SDN controller calculates latency-compliant end-to-end paths to edge servers for task offloading and execution, potentially spanning multiple domains. From these latency-compliant paths, the controller selects the edge server that minimizes task processing costs. The selection of the optimal path-server pair is a dynamic decision and depends on the network state within each domain, the overall logical network view maintained by the domain SDN controller, as well as the network operator’s objectives. For example, the network operator may prioritize selecting cost-effective servers, even if it requires sacrificing latency for a short time. In this work, we aim to minimize the overall network cost by reinforcing the SDN controller to get synchronized by the most important neighboring SDN controllers for maintaining as close to an up-to-date network view as possible. By maintaining this network view, the controller can most of the time select the most cost-efficient edge servers for multiple tasks, while ensuring that the latency constraints are satisfied.

D. Controller Synchronization Problem

The main questions we aim to answer in this work are the following: *What synchronization policy should we implement for the distributed controllers, based on the ”AR/VR Optimization” SDN application, to optimize our network performance metric over a long-term period? Additionally, how frequently should the controllers be synchronized to achieve satisfactory long-term performance?* These questions are challenging to address with conventional optimization algorithms due to the unpredictable timing of user request arrivals, the stochastic nature of the network state, and the limited inter-domain state information exchanged between controllers. Therefore, in this work, we apply model-free RL approaches to develop a robust synchronization policy that adaptively manages the synchronization frequency of distributed SDN controllers and selects the optimal subset of controllers to synchronize during each time period, thereby optimizing the defined performance metrics. Finally, we assume that the synchronization policy is deployed in one of the SDN controllers, as proposed by the authors in [5]. This controller is tasked with two main functions: (1) formulating synchronization policies, and (2) aggregating domain network states from the derived synchronization policy to update its global network view.

III. Problem Formulation

In this distributed SDN environment, our goal is to develop an intelligent synchronization policy that will be deployed in one of the distributed SDN controllers for maintaining an up-to-date global network view and thus achieving long term optimization of the described performance metric. Figure 1 illustrates a distributed SDN architecture, in which the synchronization policy is deployed in one of the controllers (C_1). This controller, based on the output of the synchronization policy, receives domain state information from a limited number of neighboring controllers and updates its network view. Before delving into the mathematical formulation, we define the unit that quantizes the synchronizable domain information and the synchronization budget.

Definition 1. *A synchronization control message (SCM) encapsulates the intra-domain topology, intra- and gateway link delays, and edge server costs. It reflects the synchronizable domain information that needs to be exchanged between domain controllers.*

Definition 2. *The maximum number of SCMs that can be exchanged between SDN controllers in a time period is defined as the synchronization budget (SB).*

Mathematical Formulation

Consider a network scenario as depicted in Figure 1, where multiple distributed controllers manage N network domains. The synchronization policy is deployed in the C_1 controller, where all user tasks K are stored in its queue. The number of domains from which the C_1 controller can receive network state updates for reconstructing its global network view is constrained by its synchronization budget $|SB|$. For each task, the C_1 controller computes an end-to-end path p from a set of available paths P and selects an edge server e from a set of available edge servers E. Each selected path $p \in P$ comprises l intra-domain and r inter-domain links, while each selected edge server incurs a cost c_e for processing the task. Let $x_i(p)$ and $z_j(p)$ denote the delays for the i-th intra-domain and j-th inter-domain links on path p, respectively. Additionally, all the user tasks have a predefined latency requirement denoted as L. The objective of the SDN controller is to minimize the total network costs by strategically offloading user tasks to a set of optimal edge servers $V^* \subseteq E$ while satisfying the latency constraints based on its inconsistent network view. This optimization problem can be formulated as follows:

$$V^* = \arg\min_{V \subseteq E} \sum_{k=1}^{K} c_e(k) \tag{1}$$

$$\text{subject to:} \quad \sum_{i=1}^{l} x_i(p) + \sum_{j=1}^{r} z_j(p) \leq L, \quad \forall k \in K. \tag{2}$$

The optimal set V^* of edge servers depends on the number of SCMs that are exchanged between the distributed controllers. Therefore, the SDN controller C_1 needs to be constantly updated by receiving the most important information from only a subset of neighboring domains to minimize the overall network cost. The decision-making process for selecting the optimal paths and set of edge servers becomes even more complex when considering additional factors such as the parallel execution of numerous tasks across various SDN controllers, the limited SB, and the network operator objectives. Given these complexities, conventional optimization techniques may prove inadequate for achieving real-time, efficient solutions. Consequently, we explore the application of various RL algorithms to dynamically and intelligently adapt to these rapidly evolving conditions.

A. MDP Formulation

We formulate our controller synchronization problem as a MDP [11] with the three tuple (S, A, R) considering N distributed SDN controllers as follows:

- S is a finite state space, represented by a one-dimensional vector $\mathrm{s} \in \mathbb{R}^N$, and denotes the number of time periods elapsed since the last synchronization of each SDN controller. This representation enables us to monitor the frequency of updates from the controllers, providing insights into the synchronization policy and the network's dynamics.
- A is a finite action space, represented by a one-dimensional vector $a \in \{0,1\}^N$, where each element a_i corresponds to the decision made for the i-th controller. Specifically, $a_i = 1$ indicates that the i-th controller's SCM is to be exchanged during the current time step, while $a_i = 0$ shows that the i-th controller's SCM will not be exchanged.
- R denotes the immediate reward function for state-action pairs, expressed as $R(s,a)$. In our defined SDN application, the reward function $R(s,a)$ evaluates the action a taken at state s and it yields a positive reward when actions jointly satisfy both latency and cost constraints. Otherwise, a negative reward is imposed if either criterion is not met individually. Intuitively, our reward function provides valuable feedback to the agent for synchronizing the subset of SDN controllers with the most long-term influence on network performance.

B. Reinforcement Learning Formulation

In RL, an agent sequentially interacts with an environment, making decisions, and receiving feedback in the form of rewards or penalties. The agent's objective is to develop a policy for selecting actions based on its current state, to maximize the cumulative reward over time. The long-term reward is defined as the discounted sum of the expected immediate rewards of all future state-action pairs from the current state. This learning process is dynamic and adaptive, with the agent continually refining its policy based on ongoing interactions and feedback from the environment. The primary goal of applying RL to our problem is to approximate the optimal synchronization policy, by synchronizing the most important SDN controllers to maximize the long term reward. Therefore, starting from an initial state s_0, the goal is to find a sequence of actions within a finite horizon so that the long-term accumulated reward is maximized and thus the SDN controller will correctly offload tasks to latency compliant paths that will minimize the network operator costs. In this work, we considered both value-based and policy-based RL approaches, to address the challenges in our synchronization problem. Due to the limited space, we will only provide the high level ideas of these approaches. For readers interested in more detailed information on the implemented RL approaches and their training algorithms, further details can be found here [12] - [16].

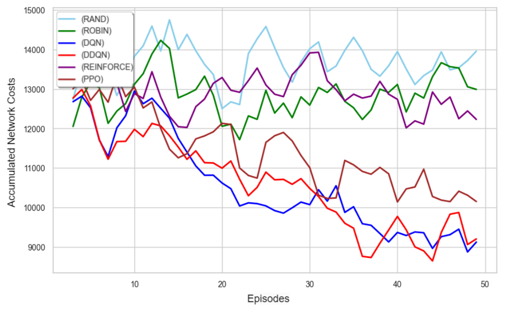

Fig. 2. The graph highlights the effectiveness and convergence behavior of each algorithm in minimizing network costs.

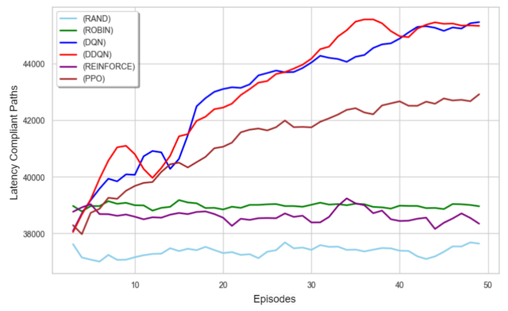

Fig. 3. The graph highlights the effectiveness and convergence behavior of each algorithm in maximizing the number of latency-compliant paths.

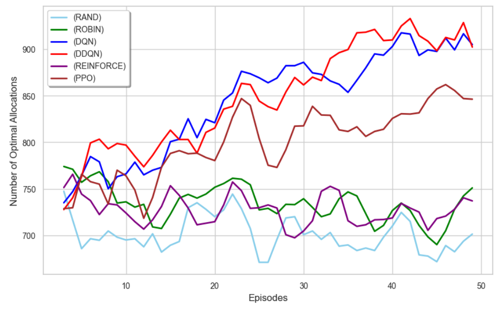

Fig. 4. The graph highlights the effectiveness and convergence behavior of each algorithm in maximizing the number of optimal server allocations for task offloading.

In value-based RL approaches, the primary objective is to estimate the value function $V(s)$, from which the policy is then calculated by selecting the optimal action at each state, while in policy-based RL approaches, the strategy is to approximate the stochastic policy directly instead of deriving a deterministic policy from an estimated value function. We focused on a selection of the state of the art value-based algorithms, specifically Deep Q-Networks (DQN) [12] and Double Deep Q-Networks (DDQN) [13]. Additionally, we explored various policy-based methods, including the REINFORCE algorithm [14] and the state of the art Proximal Policy Optimization (PPO) algorithm [15]. While we also considered A2C-A3C [16], a hybrid approach combining aspects of both value and policy-based methods, it was excluded from our in-depth analysis due to its similar performance as the REINFORCE.

Value Based Methods

For the value based approaches, we utilized a Deep Neural Network (DNN) as a function approximator to estimate the Q-function. The input to the DNN is the current state s of the distributed SDN environment, while the output represents the total number of possible actions for synchronizing the SDN controllers. The number of controllers to synchronize is constrained by the synchronization budget. For training, we employed a replay memory to store the agent's transitions and used Mean Squared Error (MSE) as the loss function, suitable for this regression problem. An epsilon-greedy strategy was implemented to balance exploration—choosing random controllers for synchronization—and exploitation—selecting the best-known controller synchronization action. Upon completing training, a greedy approach was adopted to select the optimal subset of controllers for synchronization at each time period.

Policy Based Methods

For the policy based approaches, we utilized a DNN as a function approximator to estimate directly the synchronization policy. The input to the DNN is the current state s of the distributed SDN environment, while the output represents the probability distribution over possible pairs of controllers to synchronize. Each implemented policy method has a different training approach that is covered in [14], [15]. Upon completing training, the synchronization policy is derived by sampling from the output distribution at each state s. Actions with higher probability will be sampled more frequently, ensuring that the most likely actions according to the learned policy are selected at each state.

IV. Evaluation

A. Performance Benchmarking

For evaluating the performance of our proposed SDN application, we implemented two heuristic policies along with the pre-described RL policies:

Random Policy: A simple controller synchronization policy that aims to randomly synchronize a subset of controllers at each time period. Specifically, with the given synchronization budget SB during a time period, only $|SB|$ $SCMs$ can be exchanged between SDN controllers. Note that this policy

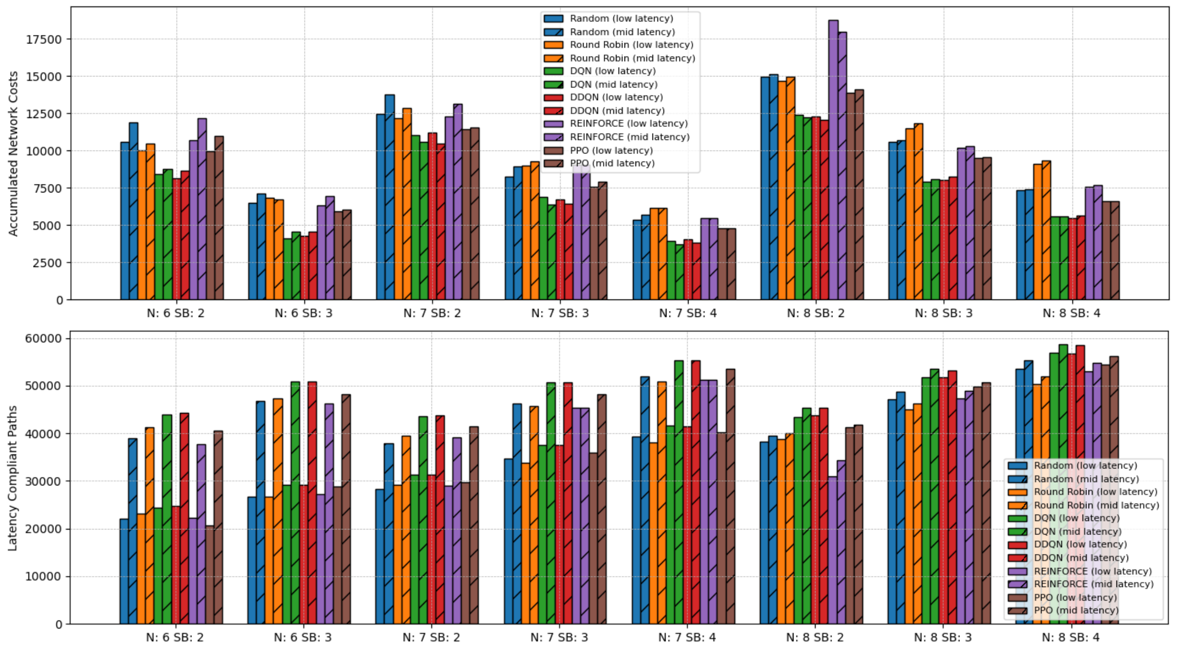

Fig. 5. Performance comparison of value-based (DQN, DDQN) and policy-based (REINFORCE, PPO) algorithms against baseline strategies (Random, Round Robin) for network configurations with varying numbers of domains (N) and synchronization budgets (SB). The upper graph shows the accumulated network costs, while the lower graph displays the total number of latency-compliant paths for two different latency requirements (low, mid) of AR/VR tasks.

was initially implemented for synchronizing distributed ONOS controllers [9].

Round Robin Policy: A structured controller synchronization policy sequentially synchronizes subsets of SDN controllers within specified time periods. Specifically, following a predetermined sequence, each subset of controllers is synchronized. At each time period, only the next subset in the sequence sends $SCMs$, adhering to the synchronization budget SB.

B. Network and RL Settings

The network topology of the distributed SDN environment was simulated using the Erdős–Rényi model [10], and we ensured that the generated graph is fully connected. Intra-domain links may experience failures and subsequently be reestablished with a probability of $p = \frac{1}{30}$ during specified time intervals. The RL algorithms and the heuristics were extensively tested in various simulated network environments with different number of network domains, data plane devices, servers, and dynamic server costs, as shown in Table I. Multiple AR/VR tasks were generated simultaneously with different latency requirements.

TABLE I
NETWORK ENVIRONMENT SETTINGS

Parameter	Value
Network Domains (N)	5 to 12
Data Plane Devices per Domain	2 to 15
Edge Servers per Domain (E)	3 to 4
Server Costs	20 to 100
Synchronization Budget (SB)	2 to 8

The RL policies were fine-tuned through extensive hyper-parameter tuning, involving variations in the number of hidden layers, the number of neurons, learning rates, and other parameters. The optimal hyper-parameters selected for deriving the synchronization policy for each case are presented in Table II. For a fair comparison of the RL policies, the same architecture (number of layers, neurons) was used across all evaluations.

TABLE II
RL PARAMETERS

Parameter	DQN	DDQN	REINFORCE	PPO
Hidden Layer Neurons	50	50	50	50
Activation Function	ReLU	ReLU	ReLU	ReLU
Batch Size	256	256	256	256
Epsilon Decay Factor	10	10	-	-
Replay Memory Capacity	40000	40000	-	-
Optimizer	Adam	Adam	SGD	Adam
Learning Rate	0.01	0.01	0.001	0.001
Discount Factor	0.1	0.1	0.1	0.1
Output Layer	Linear	Linear	Softmax	Softmax
Training Iterations	1	1	4	5
Clip Epsilon	-	-	-	0.1
Gradient Clipping Norm	-	-	-	7

C. Evaluation Results

Figures 2, 3, and 4 illustrate the evaluation performance of each implemented policy for a distributed SDN environment with $N = 10$ domains and $SB = 5$. The analysis of Figures 2, 3, and 4 demonstrates that value-based algorithms outperform policy-based algorithms, as well as Random and Round Robin synchronization policies, in minimizing network operator costs and maximizing the number of latency-compliant network paths and optimal server allocations. Figure 2 illustrates the superior efficiency and faster convergence of the DQN algorithm in reducing network operator costs, outperforming DDQN by 0.45%, PPO by 7.32%, Round Robin by 17.77%, REINFORCE by 18.37%, and Random by 22.95%. An interesting insight from the analysis is that the developed synchronization policies significantly helped the SDN controller to maximize the number of latency-compliant network paths as well as the number of optimal server allocations for minimizing the overall network cost. Figure 3 demonstrates that the DDQN is the optimal algorithm for maximizing latency-compliant network paths, with advantages of 0.19% over DQN, 4.26% over PPO, 10.45% over Round Robin, 11.53% over REINFORCE, and 15.18% over Random. Figure 4 illustrates that the DDQN algorithm excels in identifying optimal server allocations. Specifically, it outperforms the DQN by 0.28%, PPO by 5.70%, Round Robin by 15.15%, REINFORCE by 16.41% and Random by 20.61%.

As the next step, we evaluated the performance of the algorithms in various SDN environments, as detailed in Table I. In this phase, we also considered different latency requirements of the generated user tasks. We considered two scenarios: tasks with stringent low latency requirements, referred to as *low latency (10 ms)* [17], and tasks with less stringent time constraints, referred to as *mid latency (15 ms)*. Despite the extensive scope of our analysis across various network settings, Figure 5 selectively illustrates results for a representative subset of network domains (N) and synchronization budgets (SB). The upper graph in Figure 5 illustrates the accumulated network costs for various configurations of N and SB. Notably, the value-based approaches, such as DQN and DDQN, demonstrate significantly reduced network costs across all configurations, highlighting their efficiency in optimizing network operational expenses. The lower graph displays the total number of latency-compliant paths for tasks with different latency requirements. Here, DQN and DDQN again outperform other strategies, demonstrating a superior ability to maintain a higher number of latency-compliant paths. This performance is crucial for ensuring timely task offloading in time-sensitive applications. As expected, mid latency requirements enable all the policies to calculate more latency-compliant network paths, as shown in the lower part of Figure 5.

Key takeaways: Based on extensive simulations, we conclude that value-based approaches outperform policy-based methods and heuristics and even with limited synchronization (70% SB), these approaches can achieve satisfactory performance for the following reasons. Firstly, value-based methods achieve a superior exploration-exploitation balance, facilitating more comprehensive environmental learning. Secondly, they demonstrate increased robustness to dynamic network conditions, adapting more effectively to changes. Thirdly, their training processes are more stable and sample-efficient due to the incorporation of experience replay buffers.

V. Conclusion

In this paper, the limitations of existing SDN applications for time sensitive applications were addressed by proposing a novel solution to jointly satisfy latency requirements and minimize network costs. The controller synchronization problem was formulated as a MDP, and RL techniques were applied to approximate the optimal synchronization policy. Evaluation results show that value-based approaches outperform in maximizing the total number of compliant network paths while jointly minimizing network operator costs. These approaches were compared with heuristic algorithms, demonstrating superior performance.

Acknowledgment

This research was in part supported by the Army Research Office grant W911NF-23-1-0088, and the National Science Foundation CNS grant 2146838.

References

[1] Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S., “Software-defined networking: A comprehensive survey,” Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.

[2] F. Bannour, S. Souihi, and A. Mellouk, “Distributed SDN control: Survey, taxonomy, and challenges,” IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 333–354, 2018

[3] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker, “CAP for networks,” in ACM HotSDN, 2013.

[4] Poularakis, K., Qin, Q., Ma, L., Kompella, S., Leung, K. K., and Tassiulas, L., “Learning the optimal synchronization rates in distributed sdn control architectures,” in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1099–1107.

[5] Zhang, Z., Ma, L., Poularakis, K., Leung, K. K., Tucker, J., and Swami, A., “Macs: Deep reinforcement learning based sdn controller synchronization policy design,” in 2019 IEEE 27th International Conference on Network Protocols (ICNP), 2019, pp. 1–11

[6] Zhang, Z., Ma, L., Poularakis, K., Leung, K. K., and Wu, L., “Dq scheduler: Deep reinforcement learning based controller synchronization in distributed sdn,” in ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–7

[7] Mudvari, A., Poularakis, K., and Tassiulas, L., “Robust sdn synchronization in mobile networks using deep reinforcement and transfer learning,” in ICC 2023 - IEEE International Conference on Communications, 2023, pp. 1080–1085.

[8] Mudvari, and Tassiulas, L., ”Joint SDN Synchronization and Controller Placement in Wireless Networks using Deep Reinforcement Learning,” IEEE/IFIP Network Operations and Management Symposium (NOMS), 2024

[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow et al., “ONOS: Towards an open, distributed SDN OS,” in ACM HotSDN, 2014.

[10] P. Erdős and A. Rényi, ”On Random Graphs I,” Publicationes Mathematicae, vol. 6, pp. 290-297, 1959.

[11] White, C., Markov decision processes. Springer, 2001

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602, 2013.

[13] van Hasselt, H., Guez, A., and Silver, D., “Deep reinforcement learning with double q-learning,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, 2016.

[14] Williams, R. J., “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine learning, vol. 8, no. 3-4, pp. 229–256, 1992.

[15] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, ”Asynchronous methods for deep reinforcement learning,” Proceedings of the International Conference on Machine Learning (ICML), 2016

[17] European Telecommunications Standards Institute, ”ETSI TS 122 186 V16.2.0: 5G; Service requirements for enhanced V2X scenarios (3GPP TS 22.186 version 16.2.0 Release 16),” ETSI, Nov. 2020.