A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks

Original
A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks / Sacco, Alessio; Esposito, Flavio; Marchetto, Guido. - ELETTRONICO. - (2020), pp. 546-547. ((Intervento presentato al convegno 2020 16th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT) tenutosi a Virtual Event nel November 2020 [10.1145/3386367.3431670].

Availability:
This version is available at: 11583/2870934 since: 2021-07-26T10:16:18Z

Publisher:
Association for Computing Machinery (ACM)

Published
DOI:10.1145/3386367.3431670

Poster: A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks

Alessio Sacco
alessio_sacco@polito.it
Politecnico di Torino

Flavio Esposito
flavio.esposito@slu.edu
Saint Louis University

Guido Marchetto
guido.marchetto@polito.it
Politecnico di Torino

ABSTRACT

The abiding attempt of automation has also pervaded computer networks, with the ability to measure, analyze, and control themselves in an automated manner, by reacting to changes in the environment (e.g., demand) while exploiting existing flexibilities. When provided with these features, networks are often referred to as "self-driving". Network virtualization and machine learning are the drivers. In this regard, the provision and orchestration of physical or virtual resources are crucial for both Quality of Service guarantees and cost management in the edge/cloud computing ecosystem. Auto-scaling mechanisms are hence essential to effectively manage the lifecycle of network resources. In this poster, we propose `Relevant`, a distributed reinforcement learning approach to enable distributed automation for network orchestrators. Our solution aims at solving the congestion control problem within Software-Defined Network infrastructures, while being mindful of the energy consumption, helping resources to scale up and down as traffic demands fluctuate and energy optimization opportunities arise.

CCS CONCEPTS

• **Networks** → *Network algorithms*; • **Computer systems organization** → *Redundancy*; • **Computing methodologies** → *Reinforcement learning*.

KEYWORDS

reinforcement learning, self-driving networks, auto-scaling

ACM Reference Format:
Alessio Sacco, Flavio Esposito, and Guido Marchetto. 2020. Poster: A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386367.3431670

1 INTRODUCTION

Recent advantages in artificial intelligence (AI) and machine learning (ML) are paving the path to autonomous and *self-driving* networks: networks that measure, analyze and control themselves in an automated manner, reacting to changes in the environment, e.g., demand [2, 7]. One relevant tract of autonomous networks is the ability to auto-scale the resources up and down in harmony with the traffic demand. Traditional threshold-based and recent ML-based policies are often unable to address the high complexity of networks and satisfy carrier-grade requirements such as reliability and stability.

State-of-the-art solutions hardly combine these features altogether, such as [4] whose primary goal is the energy efficiency, or [6], which automatically scales Virtual Network Function instances via an ML classifier. Although reinforcement learning is emerging as a valuable technique to solve many network problems, as in [5, 8], there is no solution incorporating network information to automatically and efficiently orchestrate network resources in a distributed manner.

To this end, we present `Relevant`, a reinforcement learning approach that aims at learning how to scale without requiring any prior network knowledge. The design goal of `Relevant` is to mitigate traffic congestion while saving energy by adjusting and optimizing itself as needed. In contrast to other studies, our solution does not require any human instructions to define control policies that efficiently choose network elements to stay active while guaranteeing performance goals. Additionally, the learning process is distributed to more agents to overcome a single entity deployment's typical drawbacks. In such a way, the knowledge about the environment, needed to decide the best action, is limited yet intelligently used by the controllers. The state space of the Q-learning problem abroad each agent refers to the global network, but is obtained through an information exchange protocol rather than via a more invasive central metrics collector. In this context, the controller collects statics about the underlying network and sends the elaborated information to the peers. The distributed detection of congestion exploits a large number of SDN switches spread across

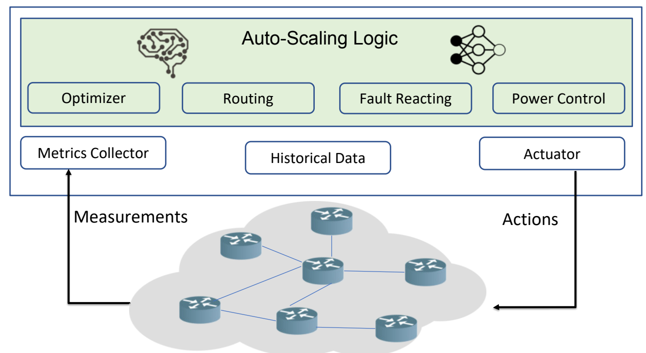

Figure 1: System overview. The (logical) network controller receives in input traffic statistics and outputs new flow routes and power on/off commands.

CoNEXT '20, December 1–4, 2020, Barcelona, Spain

ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431670

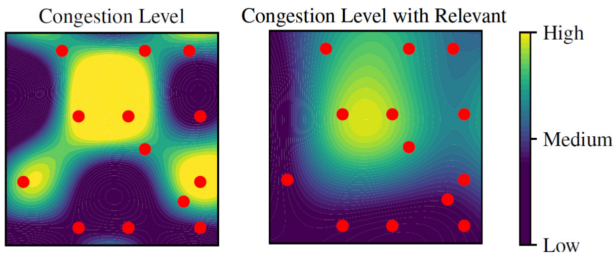

Figure 2: Heat map showing the congestion level before (left) and after (right) the application of the management layer offered by `Relevant`. Red dots indicates network nodes.

the topology as probes. This leads to a multi-agent reinforcement learning setting, which can drastically enhance the centralized strategy.

2 SYSTEM DESIGN

Our system consists of a multi-controller architecture to manage a significant amount of statistics and possible actions to implement. The controller monitors the state of each switch to detect if one of the following events occurs: the switch is overloaded (congestion), the switch in under-utilized and can be deleted (cost-saving), the switch fails, and the connectivity can be no longer guaranteed (failure). With the network partition, the controller supervises a subset of switches and communicates with the other controllers to obtain a consistent network view.

The decision logic is the fruit of a (self-)learning process, built upon the reinforcement learning (RL) framework. In our solution, each RL agent uses the one-step *Q-learning* algorithm [9]. Figure 1 shows the components and functionalities involved during the process. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The agent collects the statistics and combines them with previous historical data. Data are then passed to the optimizer module, which enforces the RL process and outputs the best possible strategy. The action set consists of a list of binary variables denoting the subset of network resources (links and nodes) to stay active. Hence, this set of active elements is larger when the traffic demand is raised, and smaller when the network is poorly utilized. According to the actions, then, routing must be re-arranged to re-distribute flows exploiting new links or bundling them to release the resources.

However, the RL agent is supposed to monitor the entire topology, which may be infeasible for vast networks. For this reason, the knowledge of the current state is spread among multiple agents that can manage the entire state space more efficiently. The distribution reduces the burden of data collection and action implementation, but enables considering a global view of the network in the scaling mechanism and the routing decisions.

We developed and implemented these functionalities in a preliminary version of the system over Mininet emulator [1] and utilizing the Ryu technology [3] for the controllers. Two controllers manage the network and activate more paths to diminish the congestion level. The advantages of `Relevant` can be observed in Figure 2, where the congestion level is shown for a traditional management approach (left) and for a network strengthened with our system (right). Red circles mark the SDN switches of the network scenario. A high congestion level (yellow) indicates that links and nodes are highly utilized and the demand is reaching the capacity. In contrast, a low level (blue) denotes mild resource usage. We can hence observe how the network can adapt to whatever the traffic does. In fact, `Relevant` prevents congestion collapse by adequately allocating the network resources that accommodate traffic demand. When the demand peak is passed, the system powers off a subset of idle components to achieve energy proportionality, still meeting the current traffic load.

Another significant functionality offered by `Relevant` is the ability to react to failures of network resources. By continuously monitoring the devices' status, the controller can detect which element is failing, and accordingly react. During this phase, the routing module computes new paths for the flow affected by the fault.

3 CONCLUSION

We proposed `Relevant`, a system that allows deploying network resources tracking the network utilization. The network agent can dynamically activate or deactivate links and nodes in an "as needed" fashion to minimize the energy consumption and the resulting costs. The decision logic is split across multiple controllers to improve the management of a large quantity of information needed for accurate actions. We expect that our system will enable both high application satisfaction and minimal management costs when deployed over more challenging environments. Future steps will include extensive evaluation of `Relevant`'s behavior in these contexts and an in-depth analysis of the learning process.

4 ACKNOWLEDGEMENT

This work has been partially supported by Comcast and by NSF awards CNS-1647084, CNS-1836906, and CNS-1908574.

REFERENCES

[1] Rogério Leão Santos De Oliveira, Christiane Marie Schweitzer, Ailton Akira Shinoda, and Ligia Rodrigues Prete. 2014. Using mininet for emulation and prototyping software-defined networks. In *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 1–6.
[2] Nick Feamster and Jennifer Rexford. 2017. Why (and how) networks should run themselves. *arXiv preprint arXiv:1710.11583* (2017).
[3] Ryu SDN Framework. 2017. *Ryu Controller*. Retrieved 2020-9-30 from https://ryu-sdn.org/
[4] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. 2010. Elastictree: Saving energy in data center networks.. In *Nsdi*, Vol. 10. 249–264.
[5] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. 50–56.
[6] Sabidur Rahman, Tanjila Ahmed, Minh Huynh, Massimo Tornatore, and Biswanath Mukherjee. 2018. Auto-scaling VNFs using machine learning to improve QoS and reduce cost. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.
[7] Alessio Sacco, Flavio Esposito, and Guido Marchetto. 2020. RoPE: An Architecture for Adaptive Data-Driven Routing Prediction at the Edge. *IEEE Transactions on Network and Service Management* 17, 2 (2020), 986–999.
[8] Pengcheng Tang, Fei Li, Wei Zhou, Weihua Hu, and Li Yang. 2015. Efficient auto-scaling approach in the telco cloud using self-learning algorithm. In *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.
[9] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.