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.

KEYWORDS
reinforcement learning, self-driving networks, auto-scaling

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.
With the network partition, the controller supervises a 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


