Machine-Learning-Assisted Routing in SDN-based Optical Networks

Sebastian Troia[(1)], Alberto Rodriguez[(1)], Ignacio Martín[(2)], José Alberto Hernández[(2)], Oscar González de Dios[(3)], Rodolfo Alvizu[(1)], Francesco Musumeci[(1)], Guido Maier[(1)]

[(1)] Politecnico di Milano, Dipartimento di Elettronica Informazione e Bioingegneria, Italy sebastian.troia@polimi.it
[(2)] Universidad Carlos III de Madrid, Telematics Engineering Department, Spain
[(3)] Telefonica Global CTO, Spain

Abstract *We demonstrate a Machine-Learning-based routing module for software-defined networks. By training with the optimal routing solutions of historical traffic traces, the module can classify traffic matrices to provide real-time routing decisions.*

Introduction

The amount of data transported by current telecommunications networks and the complexity of the applications they generate are challenging network operators, mainly due to the development and variety of network services requested by end-users. Moreover, network operators must deal with high dynamics of bandwidth requirements, so that network reconfiguration is often required to efficiently exploit the available capacity. However, performing optimal routing of several end-to-end connections is typically a very complex task, thus, efficient algorithms, such as those enabled by machine learning (ML), are needed to quickly provide accurate network reconfiguration in reasonable time.

Software-Defined Networking (SDN) has recently emerged as one of the most promising technologies for implementing centralized and programmable control planes. The SDN's logically-centralized control plane is able to acquire network information, such as network topology, topology updates, real-time bandwidth requests, link load, network device status, etc., and use them to feed optimization algorithms. Given this large number of information available from the control plane, it is possible to train machine learning algorithms to automatically and continuously optimize the network.

In this demonstration we show an SDN architecture based on ONOS controller[1] which is able to train and apply machine learning models to determine optimized network configuration upon distinct traffic matrices. To do this, we have implemented an intelligent network optimization module called Machine Learning Routing Computation (MLRC) module that drives the provisioning of paths in an SDN network. Using REST APIs[1], it captures traffic matrices with variable granularity (e.g., every 5 seconds) and trains the model continuously in order to keep it updated. The purpose of MLRC is to classify traffic matrices by means of a supervised learning algorithm trained with a set of optimal routing solutions. Such optimal routing solutions have been obtained using the Net2Plan network optimization tool[2]. Once the MLRC module has been trained, it is used to provide real-time routing decisions upon the detection of changes in the network traffic matrix.

Demo live presentation

The demonstration will be provided exploiting a virtualized lab created with Mininet[3], in the framework of EU H2020 Metro-Haul project[2]. Mininet is a network emulator that runs a collection of end-hosts, switches and links on a single Linux kernel. In Fig.1(a) we show the network topology used in the demonstration which is composed by 12 Open vSwitches[3] and 4 end hosts/servers.

The control plane is deployed with the ONOS SDN controller[1] (version 1.12). Via the Northbound interface, it provides APIs to the application plane. ONOS is a carrier-grade SDN controller that consists of applications that manage several network functions, such as: host mobility, Packet-Optical integration, proxy ARP, etc. Moreover, the Southbound interface is used by the ONOS controller to implement communication with the infrastructure plane. In this demo we use the OpenFlow[4] protocol, that gives access

[1] Weblink: https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API
[2] Weblink: https://metro-haul.eu
[3] Open vSwitch is an open-source implementation of a distributed virtual multilayer switch. weblink: *https://www.openvswitch.org/*

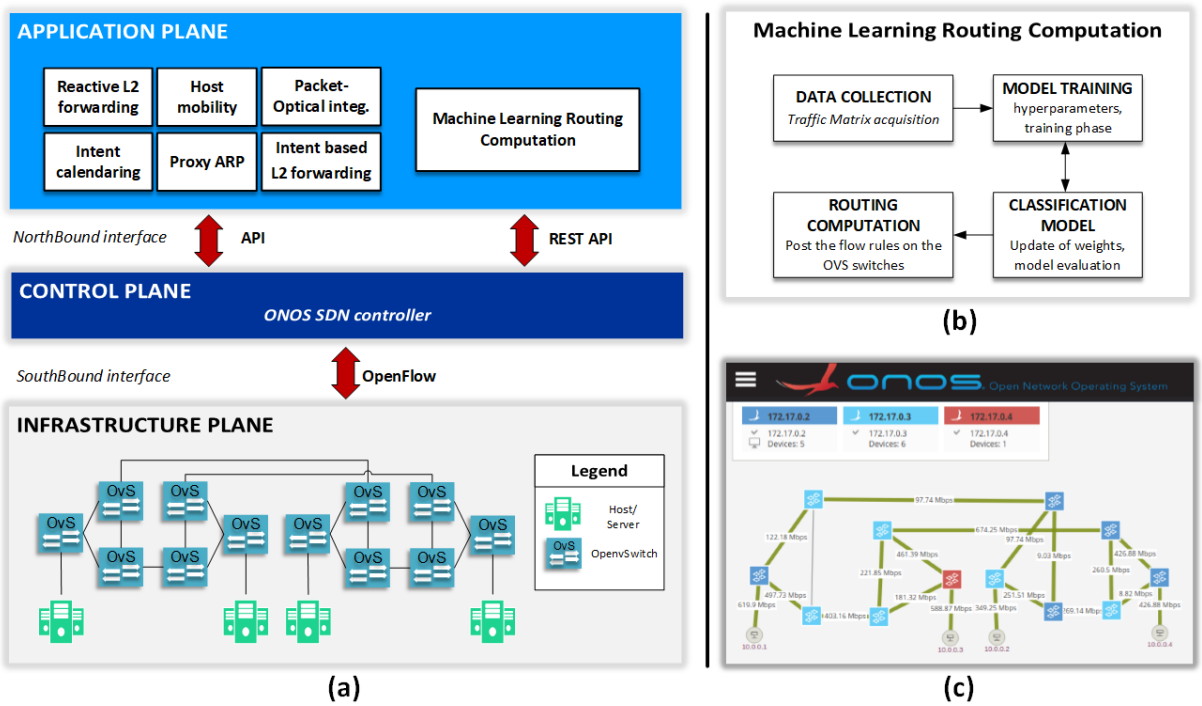

Fig. 1: Machine learning aware SDN network architecture. (a) SDN architecture composed by Infrastructure, Control and Application planes. (b) Detailed schema of the machine learning Routing Computation module. (c) GUI provided by the ONOS SDN controller in which is visible the topology and the flows of the Mininet network.

to the forwarding plane of the Open vSwitches. Furthermore, ONOS provides a graphical user interface (GUI) from which the network topology and installed routing paths can be observed (see Fig.1(c)).

Traffic is generated by the 4 hosts using the Distributed Internet Traffic Generator (D-ITG) tool[5]. D-ITG is a platform capable of producing packet-based traffic, emulating various stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random variables (e.g., with Exponential, Uniform, Cauchy, Normal, Pareto, etc.). D-ITG supports both IPv4 and IPv6 traffic generation and it is capable to generate traffic at network, transport, and application layer.

To demonstrate the performance of dynamic reconfiguration of routing plan given by the machine learning module, we run several experiments on the described testbed. Assuming that the ML model is continuously trained with the traffic matrices extracted from the network, we designed two use cases.

The first one uses generic TCP traffic to generate all possible combination of flows between the available hosts. Each host acts as a server for the other three and, meanwhile, it is client of each server instantiated on the others. While the active cycles follow an Exponential distribution, the distribution for the inter-departure time is set to Poisson.

The second use case corresponds to a subtler scenario. TCP connections share the network with UDP flows that have higher QoS requirements than the former. This example propounds Video streaming and VoIP flows to take this role. VoIP profile is already integrated inside D-ITG using the codecs of G.711 and G.729 families whereas video streaming traffic can be replicated by fine tuning the traffic generation options exposed by D-ITG.

The goal of the use cases is to show two fundamental aspects of the demo:

1. **Show in real-time the learning process of the ML model**. Traffic flows are collected from the MLRC module in order to generate a $4x4$ traffic matrix with the end-to-end instantaneous traffic. After an initial collection of about $10,000$ traffic matrices, the ML model is trained in order to generate the appropriate path for each flow.
2. **Show what the ML model has learned**. Once the model has been trained with traffic traces derived from the traffic profiles in the two use cases, it is compared with the native ONOS app, called *Reactive L2 forwarding* (see Fig.1(a)). This application provides always the shortest path between two end hosts. The MLRC app instead provides a smarter routing scheme by decreasing network congestion.

During the ECOC demo session, we will show how the ML model provides a routing strategy that is different from the one provided by the native application of ONOS Reactive L2 forwarding. In order to reduce network congestion, the MLRC module will propose routing that does not correspond to shorter ones, thus avoiding bottlenecks and congestion in advance.

In the next section, we will show in more detail the ML model implemented in this demo. Furthermore, we will show that the time required to compute the routing scheme and the consequent installation of the flow rules takes less than 100 ms.

Machine Learning Routing Computation

We now provide a more detailed description of the MLRC module, which consists of 4 sub-modules: Data collection, Model training, Classification model and Routing computation, as shown in Fig.1(b).

Data collection. This is the first phase of the module, that is, the acquisition of traffic matrices from the network. Every 5 seconds, the amount of bytes on each traffic flow is extracted from the switches and are subsequently stored. After that, they are made available to the next sub-module.

Model training. In this sub-module the machine learning algorithm is trained. This represents the heart of the whole proposed system. From a pure machine learning point of view, we thought of routing as a classification problem. Each traffic matrix has its own optimal routing configuration, obtained through Net2Plan[4] [2], that can be shared by different traffic matrices. The goal of this sub-module is to learn how to classify traffic matrices that share the same routing configuration. For this demonstration, we adopt a *logistic regression* classifier, due to its simplicity and explainability.

Classification model. This sub-module hosts the last updated classifier that is actually used to classify the input traffic matrix. The output of this sub-module is the optimal routing scheme that is passed to the *Routing computation* module. The subdivision between *Model training* and *Classification model* was made to have a scalable model, as it is possible to add other classification algorithms, and always updated, as the training is done continuously.

[4]For each traffic matrix an optimization problem is solved which aims to minimize network congestion. To obtain this result, we used Net2Plan, a network optimization tool, thanks to which it is possible to label traffic matrices, i.e. to associate them with routing configurations.

Routing computation. Finally, the routing scheme obtained by the classifier is appropriately translated into flow rules for network switches. After that it overwrites the old flows with those just obtained, avoiding memory over-flow of the switches.

ML-based routing improves the shortest path based algorithms, because in addition to proposing a routing scheme based on traffic history, it minimizes network congestion in a dynamic way. Furthermore, the machine learning module takes about 80 ms to acquire the traffic matrix, obtain the routing configuration, and install the flow rules, enabling real-time network re-configuration.

Conclusions

We demonstrate the use of Machine Learning to provide dynamic network routing configuration that can be integrated with SDN and fulfil the requirements of network operators. As a result, the ML model can be optimally mapped into ONOS SDN controller and applied to our SDN-testbed network in such a way that with any incoming change in the traffic matrix, the SDN network is capable to recompute its routing configuration and apply it in a very small lapse of time. In spite of this simple setting, industrial applications can leverage more advanced models together with real traffic matrices coming from real networks.

Acknowledgment

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul project.

References

[1] P. Berde et al., “ONOS: towards an open, distributed SDN OS. In Proceedings of the third workshop on Hot topics in software defined networking,” ACM pp. 1-6 (2014).

[2] P. Pavon-Marino et al., “Net2plan: an open source network planning tool for bridging the gap between academia and industry,” IEEE Network, vol. **29** no 5 p. 90-96 (2015).

[3] N. Handigol et al., “Reproducible network experiments using container-based emulation,” In Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 253-264 ACM (2012).

[4] N. McKeown et al., “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review 38.2 pp. 69-74 (2008).

[5] A. Botta et al., “A tool for the generation of realistic network workload for emerging networking scenarios,” Computer Networks (Elsevier), Vol. **56**, Issue 15, pp 3531-3547 (2012).