Enabling external routing logic in ONOS with Intent Monitor and Reroute service

Davide Sanvito, Daniele Moro, Mattia Gulli*, Ilario Filippini, Antonio Capone
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy
{name.surname}@polimi.it - *mattia.gulli@mail.polimi.it

Andrea Campanella
Open Networking Foundation
Menlo Park, CA, USA
andrea@opennetworking.org

***Abstract*—SDN intent-based networking allows programmers to specify high-level policies without worrying about the low-level configurations which are compiled by the controller. Our Intent Monitor and Reroute service (IMR) extends the capabilities of ONOS Intent Framework by allowing to both compile multiple intents together and to re-optimize their paths, according to the network state and their flow statistics, via off-platform applications (OPA). In this demo we show two examples of intent-based ONOS application whose network performance can be improved, with no modifications to their code, exploiting the IMR service.**

I. Introduction

Software-Defined Networking (SDN) makes computer networks more programmable and flexible to manage thanks to a common open programming interface and a match-action abstraction: processing of packets is done through a pipeline of tables enabling the application running on the controller to specialize each device function according to the specified set of packet header fields and corresponding actions.

Intent-based networking is a recent SDN paradigm where application developers can specify a high-level policy without providing low-level details to implement it in the network. Popular SDN controllers, such as ONOS [1], offer a North-Bound Interface (NBI) to submit intents. The intents get translated by the controller, via a compilation process, to low-level flow rules to achieve the requested objectives. In case of ONOS, the Intent Framework is the subsystem designed for such compilation task.

One of the most interesting aspects of intent-based networking is that the programmer is relieved from handling the changes in the underlying network. In case of network failures, for example, the controller will transparently re-compile the intent and accordingly re-configure the network to restore the requested connectivity with no intervention from the application or the user. One of the main limitations of the Intent Framework is that each connectivity intent gets individually compiled to one of the shortest paths, eventually including constraints about a subset of network devices to be traversed or a required amount of bandwidth.

This demonstration shows two possible use cases for the ONOS Intent Monitor and Reroute (IMR) service we presented in the main conference [2]. IMR is a new service which extends ONOS Intent Framework's capabilities to jointly consider multiple intents during the compilation and to make it able to re-actively adapt the routing not only according to topology changes, but also based on flow-level statistics events enabling, for example, the optimization of a global network objective such as minimizing the Maximum Link Utilization (MLU). ONOS applications and users can specify, through the IMR service, a set of intents whose statistics are kept monitored and exposed to an external routing logic. This allows to integrate an external plug&play Traffic Engineering (T.E.) logic, implemented as an Off-Platform Application (OPA), which can be transparently re-used by any intent-based application.

978-1-5386-4633-5/18/$31.00

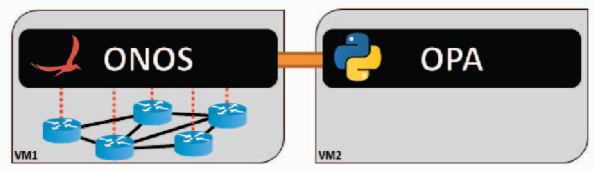

Fig. 1: Demo setup

II. Description of the demonstration

In this demo we demonstrate two examples of ONOS applications, based on intents, whose network performance can be improved by exploiting the new ONOS IMR service with almost no modifications to their application code.

The setup adopted by both demo applications consists of two Ubuntu 16.04.3 (64 bit) server virtual machines (VM). As reported in Fig.1, the first VM hosts the ONOS controller and the network, emulated by Mininet [3], while the second one runs the Off-Platform Application, written in Python. The OPA and the IMR service communicate via a REST API.

In the first part of the demo we show how an user can request the monitoring and the optimization of the paths of all the intents created by the ONOS Intent Reactive Forwarding (IFWD) application, available at [4]. In this case there are no modifications to the application code and the user enables the features of IMR service through the ONOS CLI. As an external routing logic, we implemented a greedy algorithm to maximize the throughput of the flows carried by intents created by the application.

In the second part of the demo, we exploit IMR service to improve the performance of the SDN-IP application [5]. SDN-

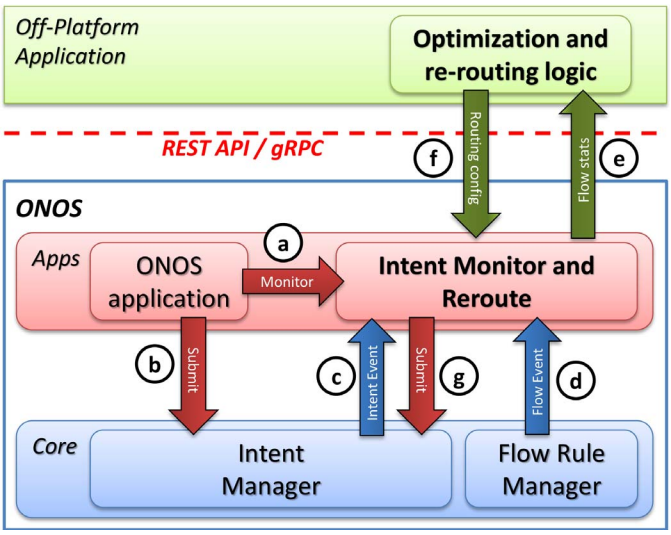

Fig. 2: IMR with ONOS and with the OPA

IP makes use of intents to carry both BGP speakers traffic and AS-to-AS traffic. In this case we want to show a more fine-grained control of IMR service by requiring, from inside the application, to optimize just the second category of intents. In this second case we selected a much more advanced external routing logic, based on optimization tools, to minimize the Maximum Link Utilization (MLU) of the network.

III. ONOS Intent Monitor and Reroute service

The Intent Monitor and Reroute (IMR) service implements a new ONOS service that enables the routing optimization for any ONOS application based on intents with minimal or no modifications to their code. As pointed out in [2], in order to avoid negatively impacting the performance of ONOS, no optimization tool is run by our IMR service in the same machine running one of the controller instances. The optimized joint re-compilation of intents is executed by an external off-platform application (OPA) and the IMR service is in charge of orchestrating the monitoring and rerouting of the intents leveraging ONOS's REST APIs or gRPC to communicate with the OPA. An ONOS application that wants to exploit features of IMR needs just to submit the intent key to it and then all the work is left to the OPA. The OPA collects the statistics, apply its optimization algorithm and re-route the intents. The details of the interactions between the IMR service, ONOS and the OPA, as depicted in Fig. 2, can be found in [2].

IV. Application examples

The following two subsections describe how IMR service can improve the performance of intent-based ONOS applications with no modifications to their code.

A. IFWD ONOS application

Intent Reactive Forwarding (IFWD) is an ONOS application [4] which reactively provides connectivity between hosts. The application installs an intent for every packet-in event that arrives at the controller. In the demo we will create a simple Mininet topology and make IFWD create a pair of intents to carry two iperf sessions between hosts. We will then require the IMR service to monitor their statistics and expose the data to the OPA which in turn will re-reroute the paths.

The functionalities offered by the IMR service can be required in two ways. In the former, an user can execute a CLI command to start the monitoring of a specific (or any) intent submitted by a given application without modifying a single line of application code, while in the latter the application explicitly invokes `startMonitorIntent(intentKey)` IMR API for each intent to be monitored. The first part of the demo adopts the first approach.

As a first example of OPA we developed a Python application [6] which retrieves a single instance of a traffic matrix (TM) through IMR's REST API and then applies a simple greedy algorithm that iteratively selects for each demand (sorted in descending order by rate) the shortest path on the capacitated residual graph. The computed paths are enforced into the network through IMR's REST API. In this case the OPA is trying to maximize the total throughput of the iperf sessions.

Even if the OPA is able to reroute intents via IMR, it does not limit the effectiveness of the Intent Framework in recovering from failures [7]: in case of failures, if an alternative path to restore the connectivity is available, the path enforced by the OPA is quickly overridden by ONOS.

The code implemented by the OPA is available as open source at [8] together with the instructions to reproduce the experiments.

A trivial extension for the OPA logic consists in iterating the approach to periodically collect traffic matrix data and re-optimize the paths which realize the application intents. We can define as training interval a set of consecutive polling cycles. After each training interval, the OPA builds the worst case TM (by keeping for each demand its worst-case value occurred in the latest training interval) and then applies the same logic explained above to maximize the throughput.

It should be noted that the decoupling of the OPA from the application which submits the intents allows to seamlessly reuse the same re-routing logic for different ONOS applications. In the next subsection we present a much more advanced re-routing logic which makes use of optimization tools to minimize the average MLU of the network and no one prevents us to use the greedy OPA with the SDN-IP application or to apply the advanced OPA to optimize the intents submitted by the IFWD application.

B. SDN-IP ONOS application

SDN-IP [5] is an ONOS application which enables a SDN network to connect to legacy IP networks using the standard Border Gateway Protocol (BGP). The SDN network exchanges routing information via eBGP and provides connectivity between external networks, behaving, from the outside, as a traditional Autonomous Systems (AS). Internally, however, the network is made of a set of OpenFlow (OF) [9] devices controlled by ONOS and a set of internal BGP speakers which communicate via iBGP among themselves and with the SDN-IP

application running on the controller. The SDN-IP application makes the controller behave as a passive BGP speaker.

SDN-IP heavily adopts intents: both the connections between the external BGP speakers and the internal BGP nodes and the connectivity between different AS are realized by means of intents. In particular, a set of Point To Point intents between external and internal BGP speakers guarantees the connectivity for eBGP peering sessions, while Multi Point To Single Point intents enable connectivity between peering interfaces. Relying on intents allows to make ONOS automatically preserve connectivity of both BGP sessions and transit traffic between AS, with no efforts from the application.

We modified the SDN-IP application to explicitly require the IMR service, via its API, to monitor all the intents related to transit traffic. We adapted the application to submit Point To Point intents instead of Multi Point To Single Point intents since IMR does not currently support them[1]. The extended application is available at [10].

The OPA might potentially keep re-optimizing the routing configuration every time it gets an updated measure of the current traffic matrix. However that specific routing is strictly tied for the traffic scenario just measured and might not be a good choice for subsequent network conditions. In addition, changing the routing too often can create instabilities into the network. As an alternative, keeping a single routing to be proactively installed for a given amount of time (e.g. one training interval) makes the network more stable at the expense of worse routing optimality.

The OPA we selected for this second demo application is Clustered Robust Routing (CRR) [11], an adaptive Robust Traffic Engineering algorithm proposed by some of the authors. CRR let the user tune the trade off between completely dynamic and completely stable routing. An optimization model is fed with the historical TM data collected over a training period. At the end of the period, by exploiting the quasi-periodicity of the traffic (under some time scales, for example 1 day), the algorithm computes a set of routing configuration to be proactively applied to the network during the following period. Routing configurations are computed to be robust against traffic deviations, with respect to the traffic experienced in training period, by considering sub-regions of traffic matrix space.

CRR defines two optimization models: the former computes robust routing configurations over a set of traffic matrices and the latter clusterizes a set of traffic matrices in the space, time and routing domain. [11] includes more details about the optimization models, together with their formulations.

In the demo we will create an emulated Mininet network replicating the Abilene [12] backbone topology and, by attaching an external BGP speaker with a single host to each node, replayed a subset of 3-days TM data using *iperf* traffic generation tool. In particular we will replay each Abilene data sample (which corresponds to the average bitrate of 5 minutes) for 15 seconds and we will monitor the Maximum Link Utilization (MLU) for each measurement round and its average over the training period. We will then compare the legacy SDN-IP application performance with the extended one, enhanced by the CRR algorithm.

The CRR OPA has been implemented as a Python application which collects TMs data from ONOS via IMR's REST API for each training period, solves the two optimization models using Gurobi [13] solver and finally schedules the activation of the robust routing configurations and applies them via IMR's REST API. After the end of the first training period we apply the routing configurations just computed and we further collect training data to reiterate the approach for the following periods.

Even if the robust nature of routing configurations allows to cope with small traffic profile variations, link failures might disrupt the connectivity of the intents. As shown in the first part of the demo, however, the *soft* nature of the constraints used by the IMR service still allows the Intent Framework to effectively recover from failures [7].

[1]This modification, however, does not affect the validity of our approach and it's in our plans to extend IMR to include the support for all Connectivity intents.

V. Conclusion

In this demo we presented Intent Monitor and Reroute service, a new ONOS module to optimize the forwarding of any ONOS application based on intents, via an external plug&play Traffic Engineering logic, with very few modifications to the application code. We demonstrated two examples of possible OPA modules which can be seamlessly reused by other ONOS applications. IMR service opens up several possibilities for OPAs which can range from classical optimization tools to novel Machine Learning or Artificial Intelligence approaches based on collected statistics. Finally, IMR might be used as pre-filtering stage for statistics collection by an OPA for purposes other than pure routing, such as monitoring and analytics.

References

[1] P. Berde *et al.*, "ONOS: towards an open, distributed SDN OS," in *ACM HotSDN*, 2014.

[2] D. Sanvito *et al.*, "ONOS Intent Monitor and Reroute service: enabling plug&play routing logic," in *IEEE NetSoft*, 2018.

[3] B. Lantz *et al.*, "A network in a laptop: Rapid prototyping for software-defined networks," in *ACM Hotnets-IX*, 2010.

[4] "ONOS Sample Applications," https://github.com/opennetworkinglab/onos-app-samples.

[5] P. Lin *et al.*, "Seamless interworking of SDN and IP," in *ACM SIGCOMM CCR*, 2013.

[6] "GitHub repository," https://github.com/ANTLab-polimi/onos-opa-example.

[7] "ONOS Wiki: Intent Framework," https://wiki.onosproject.org/x/XgAZ.

[8] "ONOS Wiki: IMR - Intent Monitor and Reroute service," https://wiki.onosproject.org/x/hoQgAQ.

[9] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, 2008.

[10] "GitHub repository," https://github.com/ANTLab-polimi/onos/tree/imr-polimi-sdnip-netsoft2k18.

[11] D. Sanvito *et al.*, "Adaptive Robust Traffic Engineering in Software Defined Networks," in *IFIP Networking*, 2018.

[12] A. Lakhina *et al.*, "Structural analysis of network traffic flows," in *ACM SIGMETRICS*, 2004.

[13] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com