Demonstrating a Distributed and Version-agnostic OpenFlow Slicing Mechanism*

Daniel Depaoli†, Roberto Doriguzzi-Corin‡, Matteo Gerola‡, Elio Salvadori‡
† University of Trento, Trento, ITALY, Email: daniel.depaoli@studenti.unitn.it
‡ CREATE-NET, ITALY, Email: {name.surname}@create-net.org

***Abstract*—Several virtualization frameworks have been proposed in the last few years for Software Defined Networks (SDN); however, they are either based on proxy-based solutions that raises scalability and robustness issues (FlowVisor), or they leverage on a simplified view of the data-path (generally based on Open vSwitch instances) that have little chances to be adopted in production network settings.**
In our demonstration we present preliminary results obtained by deploying and using a novel OpenFlow-based network virtualization mechanism. The mechanism is based on a recently proposed distributed virtualization architecture [2] that is able to run on multi-version OpenFlow scenarios.

Keywords*—*Network Virtualization, OpenFlow, FlowVisor, Software Defined Network, Demonstration

I. Introduction

Considering the adoption of the SDN paradigm at all network segment levels, the vision of a network infrastructure that can be safely shared among several users is finally becoming a reality. However, there is no common view on how an SDN network should be virtualized and, in fact, many virtualization frameworks based on SDN have been proposed recently, each one of them with their own advantages and disadvantages.

Thanks to its wide adoption and success, OpenFlow [3] is the protocol used in most of the SDN deployments and in which SDN virtualization techniques has been focused on. Leveraging on OpenFlow protocol, we may envision two major approaches to introduce network virtualization in an SDN network: (i) frameworks that leverage on an external proxy to intercept OpenFlow control messages and assign them to different controllers according to a specified "flowspace slicing" (e.g. FlowVisor [4] and VeRTIGO [5]); (ii) frameworks that assume the capability at switch level to instantiate several instances of OpenFlow virtual switches and then assign them to different controllers like [6], [7].

The former have been widely adopted thanks to their simplicity and easy of use. However they have several limitations: i) the proxy controller constitutes a single point of failure for the control plane; ii) there is non-negligible latency overhead on the control channel due to the fact that control messages have to be encapsulated/decapsulated twice and transmitted/received via a socket; iii) their implementation is strongly OpenFlow 1.0 centric. The latter have been recently proposed to overcome these limitations, however almost all of them have been proposing solutions based on Open vSwitch (OvS) [8], a virtual software switch that is being heavily used in data-center "server-centric" scenarios but has little applicability to carrier-grade switches.

In [2], a novel OpenFlow-based network virtualization framework has been proposed that overthrows FlowVisor limitations and leverages on a recent open-source datapath project named eXtensible Datapath Daemon (xDPd) [9] available for several hardware platforms. The proposed framework is based on a robust distributed virtualization architecture that is able to run on a multi-version OpenFlow switch network scenarios via a minimal overhead, both from a performance and an operational point of view.
In this companion paper, we describe a demo whose aim is to show the benefits of the approach proposed in [2]. During the demo we will (i) show the instantiation of several virtual networks controlled by various version of the OpenFlow protocol; (ii) demonstrate the robustness of this new virtualization mechanism to failures happening at node level.

II. Architecture

In [2], the authors propose a distributed virtualization mechanism designed with the following goals: (i) avoid Single Point of Failures (SPoF) through a distributed slicing architecture, (ii) provide an OpenFlow version agnostic slicing mechanism and (iii) minimize the latency overhead caused by the slicing operations.
Distributed slicing. SPoFs are avoided through the implementation of the so-called Virtualization Agent (VA) which resides on the OpenFlow-enabled switches. Being the slicing process performed at the datapath level, no external proxies are used. Therefore, no SPoFs are created.
Protocol agnostic. The VA does not inspect the control protocol to perform the slicing process but, instead, it uses version-agnostic structures containing flow matches or actions.
Latency overhead. Differently from FlowVisor, the VA neither inspects the OpenFlow protocol nor needs to establish additional TLS connections with switches and controllers. As demonstrated in the evaluation section of [2], this means lower latency overhead on the control channel.

The Virtualization Agent operates at the datapath level and is implemented as a plugin for xDPd (in Fig. 1, a concise version of the architecture is sketched). In particular, xDPd instantiates multiple Logical Switch Instances (LSIs, virtual switches in the xDPd terminology) when different versions of the OpenFlow protocol are used on the same physical switch. Furthermore, multiple controllers using the same or different versions of the protocol are handled by the VA through the so-called *OF endpoint* encapsulated within each instantiated LSI. The OF endpoint is the module in charge of providing the

*This work is partially supported by the EU FP7 ALIEN project [1].

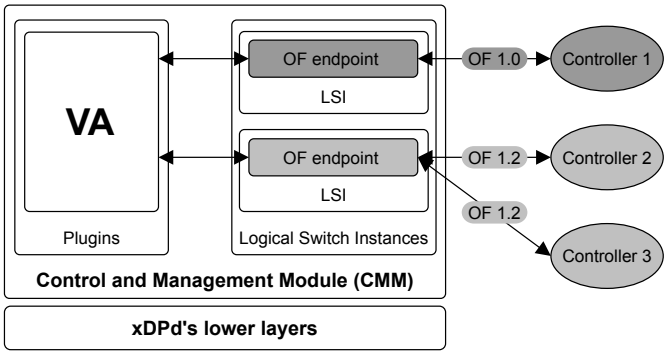

Fig. 1: The slicing mechanism is performed by the Virtualization Agent (VA) which is implemented as a plugin for xDPd.

communication interface with the controller through a specific version of the OpenFlow protocol.

III. Demonstration

The objective of the demonstration is to show how the proposed network virtualization approach, grounded on xDPd and the VA, allows: (i) multiple versions of the OpenFlow protocol to be used at the same time to control the same physical infrastructure, (ii) IPv6 and IPv4 multicast streaming experiments to be performed on different virtual networks without interfering each other and (iii) virtual networks to operate even in case of failure of one of the VA instances.

The demonstration setup, depicted as *Physical Topology* in Fig. 2, is composed of four xDPd-based forwarding nodes, each with the VA process enabled. Three commodity PCs are used to both host the virtual machines where we can generate the multimedia traffic and to run multiple OpenFlow controllers.

The VA instances running on the four switches are configured to slice the flowspace through the VLAN_ID field of the packet headers. For the demo purposes, we setup three virtual networks with three VLAN_ID values (let say 10, 20 and 30) and composed of three different subsets of the physical topology (see Slice A, B and C layers in Fig. 2). Moreover, the network interfaces of the virtual machines connected to the virtual networks are configured to inject traffic tagged with the correct VLAN_ID.
Finally, as also highlighted in Fig. 2, the multimedia traffic of Slice A is controlled by using version v1.0 of the OpenFlow protocol with *FlowMods* based on IPv4 addresses. Differently, the IPv6 multimedia traffic of Slices B and C is controlled through version 1.2 of the protocol that includes the support for matching the IPv6 source and destination addresses.

During the experiment, we will also demonstrate that the video streams flowing across two of the three slices are not perturbed by a failure of the VA process on one of the nodes. In fact, differently from other approaches with a central proxy like FlowVisor, the virtualization operations are performed directly on the nodes. Therefore, only the virtual networks including the failing node can be affected by traffic disruption. However, being the failure restricted to a single node, the controller can easily apply the necessary countermeasures by redirecting the traffic through an alternative path.

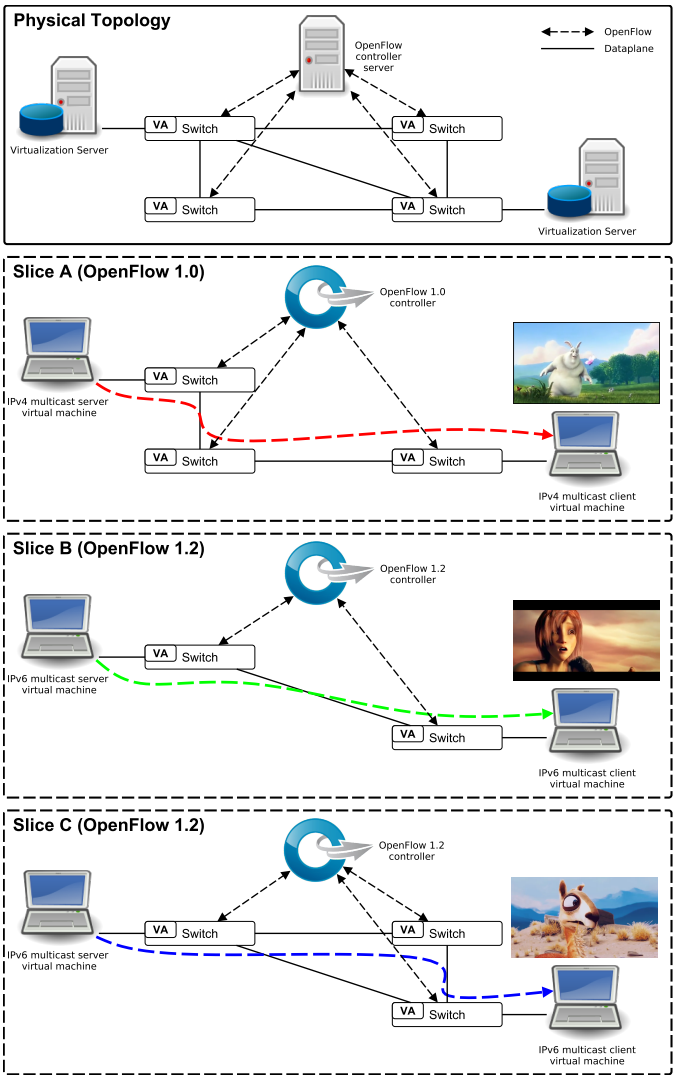

Fig. 2: Pilot setup

References

[1] "ALIEN FP7 project," http://www.fp7-alien.eu.

[2] R. Doriguzzi Corin, E. Salvadori, M. Gerola, M. Suñé, and H. Woesner, "A Datapath-centric Virtualization Mechanism for OpenFlow Networks," in *European Workshop on Software Defined Networking (EWSDN)*, 2014.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 2, pp. 69–74, April 2008.

[4] R. Sherwood et al., "Can the production network be the testbed?" in *Proc. of USENIX OSDI*, Canada, 4-6 Oct. 2010.

[5] R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "Vertigo: Network virtualization and beyond," in *European Workshop on Software Defined Networking (EWSDN)*, 2012.

[6] P. Skoldstrom and K. Yedavalli, "Network virtualization and resource allocation in openflow-based wide area networks," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 6622–6626.

[7] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, and G. Vaszkun, "Openflow virtualization framework with advanced capabilities," in *European Workshop on Software Defined Networking (EWSDN)*, 2012.

[8] "Open vSwitch website," http://openvswitch.org/.

[9] "The eXtensible Datapath daemon (xDPd)," http://www.xdpd.org/.