Enabling Network Virtualization in OpenFlow Networks through Virtual Topologies Generalization

Matteo Gerola[1]
CREATE-NET
via Alla Cascata 56/D, 38123 Povo - Trento - ITALY
Email: matteo.gerola@create-net.org

***Abstract*—A recent approach toward Network Virtualization has been proposed through FlowVisor, whose aim is to leverage on the specific features of an OpenFlow–controlled network to share the same hardware forwarding plane among multiple logical networks. In this work, an innovative system called ADVisor (ADvanced FlowVisor) which enhances FlowVisor while overcoming its major constraints is presented and a set of experimental results discussed to demonstrate its capability to provide an effective support toward a Network Virtualization architecture.**

I. Introduction

Network Virtualization (NV) is one of the most promising approaches to enable innovation in today's network. Generally speaking, NV refers to the possibility of pooling together low–level hardware and software resources belonging to a networked system into a single administrative entity. In such a way network resources could be effectively shared in a transparent way among different logical network instances corresponding to different virtual network topologies.

A recent approach toward NV has been proposed through FlowVisor [1], whose aim is to leverage on the specific features of an OpenFlow–controlled network [2] to share the same hardware forwarding plane among multiple logical networks. As highlighted by the authors in [1], one of the major limitations of FlowVisor is the inability to establish virtual topologies not restricted by the underpinning physical topology. As a consequence, FlowVisor is unable to provide researchers flexibility in designing their experiments with arbitrary network topologies on a defined physical infrastructure.

The architecture presented in Sec. II of this paper, called ADVisor (ADvanced FlowVisor), provides the functionalities to overcome the above-mentioned FlowVisor's limitation by allowing the instantiation of generalized virtual topologies in a OpenFlow network through the implementation of virtual links as aggregation of multiple physical links and OpenFlow-enabled switches.

This paper is divided in two sections: Sec. II provides an overview of the ADVisor's architecture and Sec. III shows the results of some experimental sessions.

[1]This extended abstract is an excerpt of the following paper accepted at GLOBECOM 2011: E.Salvadori, R.Doriguzzi Corin, Attilio Broglio and Matteo Gerola *"Generalizing virtual network topologies in OpenFlow-based networks"*

II. Proposed architecture

Like FlowVisor, ADVisor sits between the physical hardware and the guest OpenFlow controllers and enables the implementation of logical topologies and, like FlowVisor, ADVisor can recursively "slice" a virtual topology (see Fig. 1). Differently from FlowVisor, ADVisor does not act as a transparent proxy but can directly reply to the OpenFlow network with the purpose of enabling the instantiation of logical topologies completely decoupled from the underlying physical network.

In ADVisor, Virtual Topologies (VT) are identified through a set of tuples included in configuration files and specifying each component of a VT (virtual nodes, virtual links and virtual ports). Furthermore, the flow space of each switch in the network is partitioned among VTs through combinations of bits involving only the OSI-Layer 2 fields of the packet header such as the *VLAN ID*, the *MPLS labels* or *IEEE 802.1ad*–based multiple vlan tagging. These last two options are giving higher flexibility since they both allow the experimentation to be performed up to L2, however they are not available yet on any switch hardware being the OpenFlow specification version 1.1.0 [3] recently released.

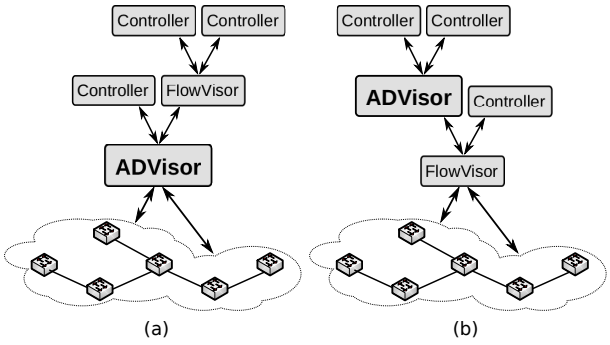

Fig. 1. (a) ADVisor can be placed between the OpenFlow controllers and the physical network or (b) between an instance of FlowVisor and the controllers to recursively "slice" a virtual topology.

As depicted in Fig. 2, ADVisor is a modular extension of FlowVisor composed of four main blocks:

Topology Monitor. This module checks the VT configuration in order to determine whether the switch that generated the OpenFlow protocol message is an end-point of a link or is part of a virtual link (e.g. switch sw2 in Fig. 3). In the first

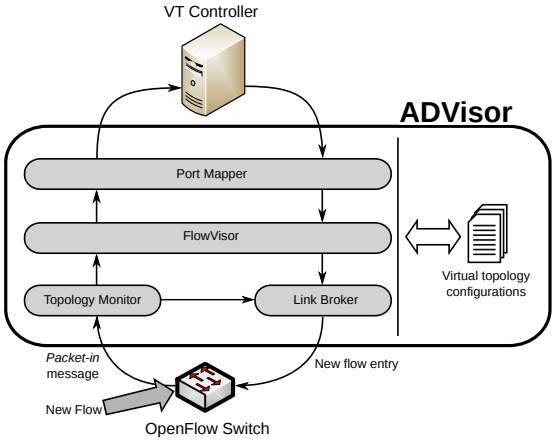

Fig. 2. ADVisor software architecture. Virtual topology configurations are stored in FlowVisor-like XML files defining all switches, links, virtual links and virtual port mappings for each VT.

case the message is forwarded to the controller, in the second case the message is managed directly by the Link Broker.

Port Mapper. This module edits the *in_port* and *actions* fields of the OpenFlow protocol messages by replacing their values with ones consistent with the virtual links configuration.

Link Broker. The Link Broker creates or modifies OpenFlow protocol messages directed to the switches. Its main objective is to control switches composing virtual links and that should not be managed by controllers. For instance, switch sw2 in Fig. 3 is a component of virtual links between switches sw1 and sw3 and between switches sw3 and sw4. OpenFlow protocol messages sent by sw2 to the controller of this topology are always managed by the Link Broker that directly replies to the switch hiding sw2 to the controller.

FlowVisor. Provides the basic slicing mechanism based on the OpenFlow protocol and manages the TLS secure connections with OpenFlow switches and controllers.

III. Evaluation results

The main goal of the experimental tests described in this Section is to show the additional overhead introduced by ADVisor to the system and how ADVisor isolates virtual topologies. Tests have been performed on a OpenFlow testbed composed of four NetFPGA-based switches (as shown in Fig. 3) and one central unit running ADVisor, FlowVisor and the NOX controller.

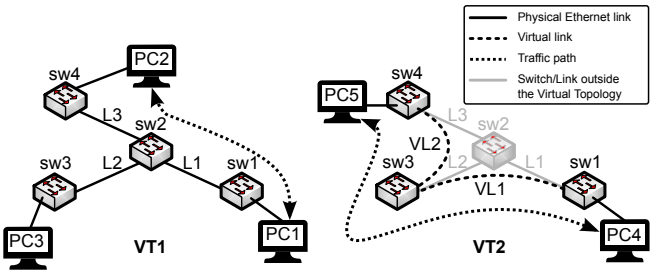

Fig. 3. The two Virtual Topologies used in the experimental tests. The figure also shows the paths traversed by the synthetic traffic during the tests.

A. Performance Overhead

This first experimentation aims at evaluating the impact of ADVisor algorithms on the operations of an OpenFlow network. More specifically, we measure and compare the latency overhead introduced on the switch-controller secure channel by both FlowVisor and ADVisor.

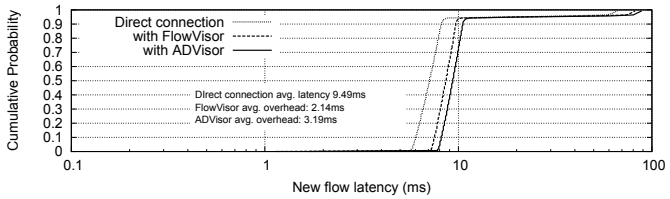

Fig. 4. CDF of virtualization overhead for new flow messages.

As shown in Fig. 4, ADVisor increases the latency by 3.19ms on average compared to a standard OpenFlow network configuration without any network virtualization mechanism (i.e. with direct connection between switch and controller), and by 1.05ms on average compared to a second scenario with FlowVisor placed between switch and controller. These values prove that ADVisor's operations do not add significant latency on the secure channel.

B. Virtual Topology isolation

The objective of this test is to prove the robustness the ADVisor's bandwidth reservation mechanism. For this experimentation we have manually limited the switches outgoing bandwidth to 10Mbps and set a minimum guaranteed bandwidth of 2Mbps for VT2.

The test is performed in two steps: (i) at time 0, 11Mbps full-duplex UDP traffic is generated between PC1 and PC2 on VT1, (ii) after 30 sec. 2Mbps full-duplex UDP traffic is generated between PC4 and PC5 on VT2.

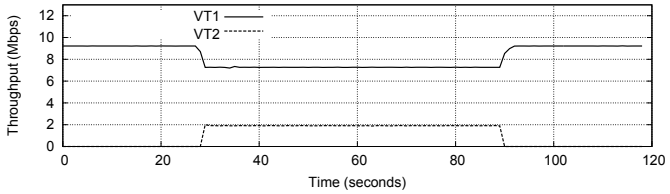

Fig. 5. Throughput measured through links L1 and L3 (see Fig. 3)

Plot in Fig. 5 shows that the isolation of virtual topology VT2 is preserved by ADVisor even though this topology is completely decoupled from the physical topology.

References

[1] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the Production Network Be the Testbed?" in *Proc. of In Operating Systems Design and Implementation - OSDI 2010*, Vancouver, BC, Canada, October 2010.

[2] 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.

[3] OpenFlow Switch Consortium, "OpenFlow Switch Specification Version 1.1.0," Tech. Rep., February 2011.