Demonstrating Generalized Virtual Topologies in an OpenFlow Network

Elio Salvadori, Roberto Doriguzzi Corin, Matteo Gerola,
Attilio Broglio, Francesco De Pellegrini
CREATE-NET
Via alla Cascata 56/D Povo, 38123 Trento - Italy
{esalvadori, rdoriguzzi, mgerola, abroglio, fdepellegrini}@create-net.org

1. 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 Network Virtualization 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 Chapter 2 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.

In this demo we will show the configuration of a simple virtual topology performed through a Web-based control framework which allows the reservation of network resources (nodes, links and bandwidth) and the management of virtual resources (virtual links and virtual ports). We will also demonstrate the effective instantiation of the virtual topology by running a synthetic traffic generator application.

Categories and Subject Descriptors

D.2.8 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Experimentation

Keywords

Network Virtualization, OpenFlow

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

2. 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 Virtual Topology (virtual nodes, virtual links and virtual ports). Furthermore, the flow space of each switch in the network is partitioned among Virtual Topologies 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.

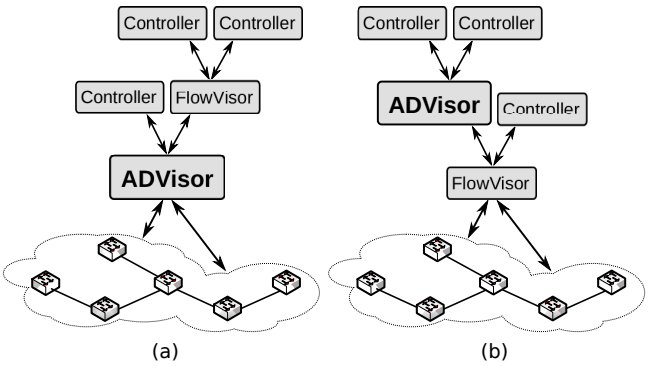

Figure 1: (a) ADVisor can be placed between the OpenFlow switches and the physical network or (b) between an instance of FlowVisor and the controllers to recursively "slice" a virtual topology.

In order to provide a full overview of the system, we will give a description of the modules composing ADVisor:

Topology Monitor. This module checks the Virtual Topology configuration in order to determine whether the switch that generated the OpenFlow protocol message is an

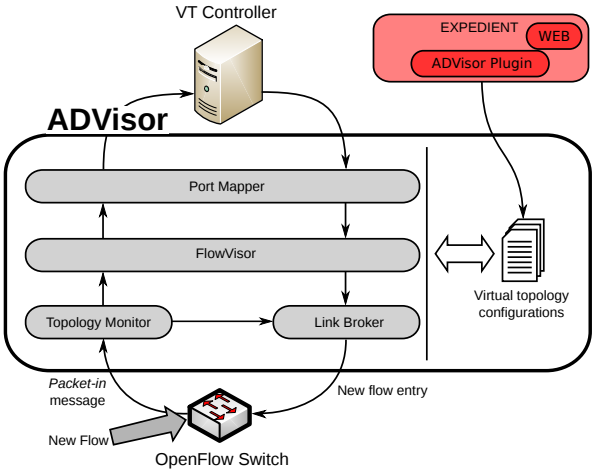

Figure 2: ADVisor software architecture.

end-point of a link or is part of a virtual link (e.g. switch sw2 in Fig. 3). In the first 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. For instance, switch sw3 in Fig. 3 is connected to the physical network through a single link while to the Virtual Topology through two different virtual links. Port Mapper remaps the in_port value contained in the Packet_in messages generated by the switch with a virtual value dependent on the virtual link.

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.

SFA-based control framework. The Expedient-based framework [4] provides an intuitive web user interface for the ADVisor managers to configure the resources for the experimenters. Expedient includes the ADVisor plugin, an extension of the original OpenFlow plugin, which allows the selection of network resources (OpenFlow switches, virtual links, bandwidth etc.) for each Virtual Topology assigned to experimenters.

3. DEMONSTRATION

The primary goal of the demonstration is to show the effective instantiation of a virtual topology and the management of virtual links and virtual ports with this initial prototype of ADVisor. The demonstration is performed on a remote testbed at the CREATE-NET premises composed of four NetFPGA-based OpenFlow switches (interconnected as shown in Fig. 3) and one central unit running ADVisor and

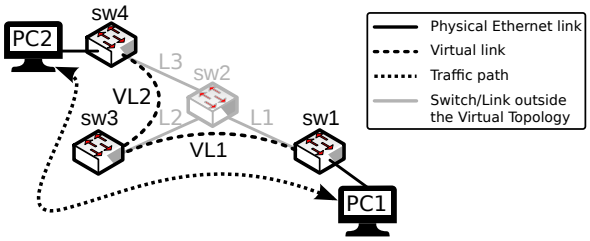

Figure 3: The Virtual Topology used in the demonstration.

an instance of the NOX controller. All testbed devices are connected through Ethernet cables and are compliant with OpenFlow specification 1.0.0. The virtual topology instantiated for the demonstration includes three switches sw1, sw3 and sw4 plus two virtual links: VL1, formed with the aggregation of physical links L1 and L2 plus switch sw2; VL2 formed with the aggregation of L2, L3 and sw2.

During the demonstration, the Expedient-based user interface (see Fig. 4) is used to configure the virtual topology depicted in Fig. 3 including virtual links VL1 and VL2; once configured, some synthetic traffic is transmitted from $PC1$ to $PC2$.

Through the output of the NOX controller, the experiment demonstrates the ability of ADVisor to correctly control switch sw2 (i.e keeping sw2 out of the view of the NOX controller) and to properly manage virtual links and virtual ports on sw3. Of course being the system fully reconfigurable, other virtual topologies "carved" from the physical topology will be shown as well.

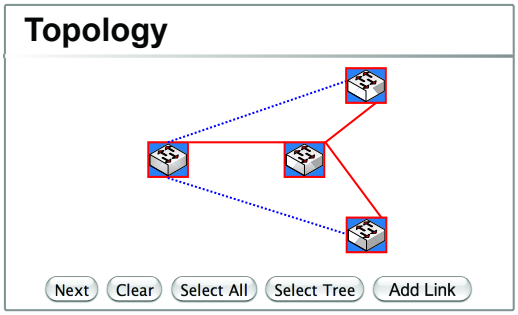

Figure 4: Screenshot of Expedient showing the topology configuration page.

4. REFERENCES

[1] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," OpenFlow Switch Consortium, Tech. Rep., October 2009.

[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.

[4] Expedient: A Pluggable Platform for GENI Control Frameworks. [Online]. Available: http://yuba.stanford.edu/~jnaous/expedient/