Automatic Configuration of Routing Control Platforms in OpenFlow Networks

Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet and Piet Demeester
Department of Information Technology (INTEC), Ghent University - iMinds
E-mail: {firstname.lastname}@intec.ugent.be

ABSTRACT

RouteFlow provides a way to run routing control platforms (e.g. Quagga) in OpenFlow networks. One of the issues of RouteFlow is that an administrator needs to devote a lot of time (typically 7 hours for 28 switches) in manual configurations. We propose and demonstrate a framework that can automatically configure RouteFlow. For this demonstration, we use an emulated pan-European topology of 28 switches. In the demonstration, we stream a video clip from a server to a remote client, and show that the video clip reaches at the remote client within 4 minutes (including the configuration time). In addition, we show automatic configuration of RouteFlow using a GUI (Graphical User Interface).

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design; Management

Keywords

Quagga; OpenFlow; Virtualization

1. INTRODUCTION

OpenFlow decouples control plane functionality from forwarding functionality of switches, and embeds it into one or more servers called controllers. In OpenFlow networks, RouteFlow [1] provides a way to run routing control platforms (e.g. Quagga). It executes switches' (OF-A, OF-B, OF-C and OF-D in Fig. 1) control logic through virtual machines (VM-A, VM-B, VM-C and VM-D in Fig. 1) which mirror a physical topology. Each virtual machine (VM) runs a routing control platform (e.g. Quagga) and is dynamically interconnected with other VMs.

Currently, configurations of RouteFlow are not automatic. Before running RouteFlow, an administrator needs to devote a lot of time in configurations: (1) creating VMs, (2) creating mapping between a VM and an OpenFlow switch, (3) creating mapping between VM interfaces and switch interfaces, and (4) writing routing configuration files (e.g. ospf.conf, zebra.conf) for each VM. For a large topology (typically for 1000 switches), it may take many days to configure RouteFlow.

SIGCOMM '13, Aug 12-16 2013, Hong Kong, China
ACM 978-1-4503-2056-6/13/08.
http://dx.doi.org/10.1145/2486001.2491695.

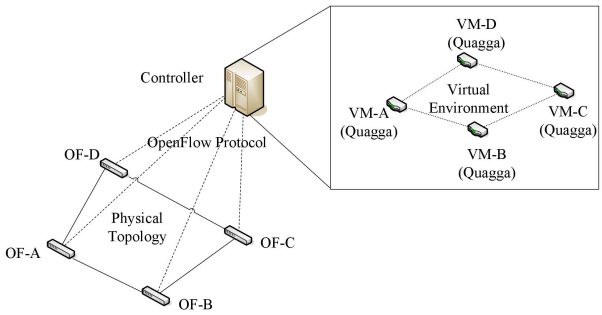

Figure 1: RouteFlow Design

We propose a framework to automatically configure RouteFlow. In our framework, we use an additional controller which runs a topology discovery module [3] to know network configurations. The network configurations are then sent to RouteFlow using configuration messages. Using these messages, RouteFlow configures itself.

For this demonstration, we use an emulated pan-European topology of 28 switches. In the demonstration, we stream a video clip from a server to a remote client. This video clip reaches at the remote client within 4 minutes (including the configuration time). This is quite optimal compared to the time consumed in manual configurations. In addition, we show automatic configuration of RouteFlow by showing configurations of VMs in a GUI.

2. AUTOMATIC CONFIGURATION OF ROUTEFLOW

In this section, we introduce our framework and present the results of the experiments performed on the OFELIA testbed [2].

Fig. 2 shows five different components of the proposed framework: (1) RF-controller, which runs RouteFlow without any manual configuration of VMs, (2) Topology controller, which contains a very small part of configurations from the administrator (e.g. a range of IP addresses for the virtual environment) and runs a topology discovery module

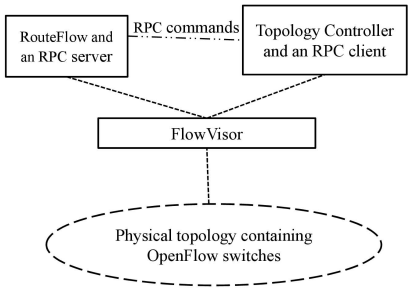

Figure 2: Framework for automatic configuration of Routeflow

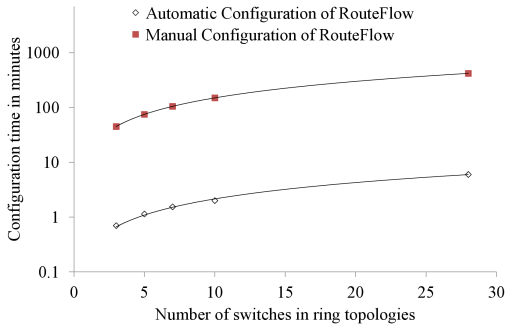

Figure 3: Configuration Time

[3] to know the network configuration (switches and links information), (3) RPC (remote procedural call) client, which collects configuration information from the topology controller and sends this to a server called RPC server, (4) RPC server, which resides in the RF-controller and configures RouteFlow on reception of configuration messages from the RPC client, (5) FlowVisor, which acts as a proxy server between a switch and controllers (the topology controller and the RF-controller in our framework).

In our framework, we used different controllers for gathering topology information (topology controller) and running RouteFlow. This is done to share the load of automatic configuration of RouteFlow.

At the initial stage, the RF-controller does not have any configurations i.e. there are no virtual machine to run Quagga. On detection of a new switch, the topology controller sends a configuration message to the RPC client, which then forwards it to the RPC server. This configuration message contains the ID of the switch and the number of switch ports. Upon receiving of the message, the RPC server creates a VM with an ID identical to the switch ID and the number of ports equivalent to the switch ports.

On detection of a new link, the topology controller computes unique IP addresses (from the range of IP addresses) for the corresponding VM interfaces, and sends this information to the RPC server through the RPC client. The RPC server then configures IP addresses of the VM interfaces.

Additionally, the RPC server writes routing configuration files (e.g. ospf.conf, zebra.conf, bgp.conf) using the information present in the configuration message sent by the RPC client.

2.1 Results of automatic configuration experiments

We perform experiments to automatically configure RouteFlow that uses OSPF (Open Shortest Path First) as a routing protocol. The experiments are performed on ring topologies with different number of switches. These topologies are generated on a node of the OFELIA testbed by using Linux processes in different network namespaces. In each Linux process, we run Open vSwitch 1.4.1 implementation [4]. Separate nodes of the testbed are used to run FlowVisor, the topology controller, and the RF-controller.

Fig. 3 shows the time of automatic and manual configurations of RouteFlow. We calculate the time in manual configurations based on personal experience. In manual configurations, we assume that the administrator takes 5 minutes in creating a VM (writing VM configurations, installing Linux distributions and packages like Quagga), 2 minutes in creating mapping between switch interfaces and VM interfaces, and 8 minutes in writing routing configurations for a VM. The figure shows that there is a large difference between automatic and manual configurations of RouteFlow.

3. DEMONSTRATION SETUP

We demonstrate the proposed framework to automatically configure RouteFlow in OpenFlow networks. For the demonstration, we connect two laptops using an Ethernet cable. The first laptop contains the RF-controller, the RPC server, the RPC client, the topology controller and the FlowVisor. The second laptop contains an emulated OpenFlow network topology, which is a pan European topology [5] consisting of 28 nodes. The clients and servers are connected with the nodes of this topology.

In the demonstration, we show automatic configuration of RouteFlow by showing switches with red and green colors in a GUI. The color of a switch remains red until it is configured by the RPC server. Otherwise, it changes to green. Note that a switch is considered as configured when it has a corresponding VM.

At the start of the experiment, we stream a video clip from a server to a remote client. At this point, there is no virtual machine present in the RF-controller. However, thanks to the proposed framework, the VMs are created and the routing protocol is enabled within a very short time, and the video clip reaches (after around 4 minutes) at the remote client.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU FP7 programme under grant agreement n^o 317576 (CityFlow) and n^o 258365 (OFELIA).

4. REFERENCES

[1] C. E. Rothenberg et al., Revisiting Routing Control Platforms with the Eyes and Muscles of Software Defined Networking, HotSDN, 2012
[2] OFELIA Testbed: http://www.fp7-ofelia.eu/
[3] Topology Discovery module: https://github.com/noxrepo/nox-classic/wiki/Discovery
[4] Open vSwitch: http://openvswitch.org/
[5] S. D. Maesschalck et al., Pan-European optical transport networks: an availability-based comparison, Photonic Network Communications, 2003