Integration of an Open-Source Software-Defined Router in the Cloud

Project partners: CENGN, TELUS and Noviflow

Revision History

Date	Version	Description
04/07/2016	1.0.0	Original Document

Contents

1. Introduction

The networking industry is going through a revolutionary change where traditional vendor-specific closed network nodes are being replaced by open and programmable software instances. Different open source projects are contributing to this evolution of networking industry by developing software packages that will eventually be able to produce carrier-grade network solutions. One of the major driving forces that has already shaped the future of network architecture is Software Defined Networking (SDN). It separates the network control-plane from the data-plane so that the underlying network fabric can be dynamically programmed.

To deploy an end-to-end network solution from open source software products, it is imperative to integrate the software elements to build a functional SDN stack. The Open Networking Foundation (ONF) [1] is a key organization that promotes the adoption of SDN through open standards development. The Atrium project [2] is the first-of-its-kind from ONF in an effort to integrate different open source SDN software to build an end-to-end network solution. In the project, an SDN-based router was built on ONOS [3] control platform using Quagga [4] software suite and a programmable switch. For the Atrium project, the Border Gateway Protocol (BGP) routing instance of Quagga software suite is used as the routing application to peer with different BGP speaking traditional routers. The router app translates the BGP routing information to OpenFlow compliant messages that is understood by the controller platform. The ONOS controller which has a global view of the network, takes instructions from the Quagga router and programs the underlying switching fabric. In the Proof of Concept (POC) demo by ONF [1], it was shown that the Atrium SDN stack was able to operate as a BGP router that peered with both software-defined and traditional BGP routers.

As part of our mission to accelerate the commercialization of emerging networking technologies, CENGN along with its Members undertakes Proof-of-concept project leveraging new products and solutions from Canadian Small and Medium Enterprise (SME) on a regular basis. TELUS, a leading telecommunication service provider in Canada as a CENGN Member selected Noviflow to create a Proof-of-concept SDN project at CENGN to understand interoperability of traditional routers and SDN routers. Project Atrium was leveraged for this TELUS-Noviflow project.

Fig. 1 shows a high-level overview of the POC testbed architecture. It consists of three different networks belonging to three different autonomous systems (ASes). Each AS runs eBGP among the peering external routers to exchange the BGP neighboring information. One of the ASes, the ASN#65000 consists of the CENGN cloud infrastructure with a software-defined Atrium router [2]. The external routers for the two traditional (non-SDN) networks (ASN#65001 and ASN#65002) consist of Junipers vMX [5] routers.

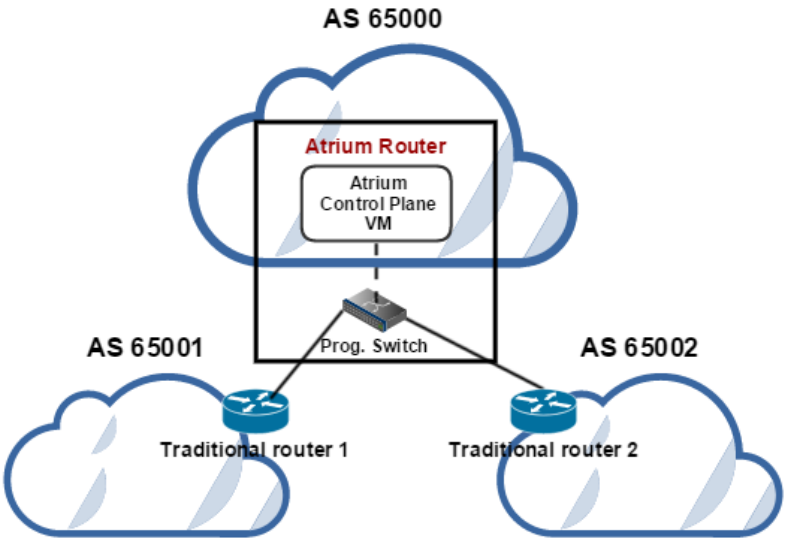

Fig. 1. BGP Peering of the Atrium router with neighboring traditional routers

The objectives of the POC testbed are:

- to integrate the software-defined Atrium router in CENGNs Mirantis OpenStack 6.1 for Juno-based cloud platform;
- to demonstrate the BGP interoperability of the SDN-based router (Atrium) with traditional routers (Junipers vMX);
- to integrate the software-defined NoviKit 250 switch as the forwarding plane device that is programmed by the ONOS controller on behest of the BGP routing application to form BGP peering with traditional routers.

This article provides the detailed architecture of the POC testbed and discusses the challenges faced and solutions devised during the testbed implementation. We hope that this article will be of great help to different vendors, operators and the open source community as a whole who plan on being early adopters of the SDN-based cloud-enabled next generation networking solutions.

2. The POC Testbed Network Architecture

Fig. 2 shows a more detailed implementation of the testbed architecture. The Atrium router is implemented as an SDN overlay on the CENGN cloud platform. The software stack of the router consists of a Forwarding Plane Manager (FPM) speaker, a router app, the ONOS controller, a device driver for the NoviKit 250 SDN switch and a Quagga host spun up in Mininet [6] which is an emulation platform for creating realistic virtual networks. This Quagga host runs a BGP daemon which is the BGP routing application of the Atrium router. The Quagga host has three interfaces; one is connected to the FPM speaker to communicate the BGP routing information with ONOS controller [3]. The other two are connected to the two neighboring AS routers vMX1 and vMX2, respectively, for

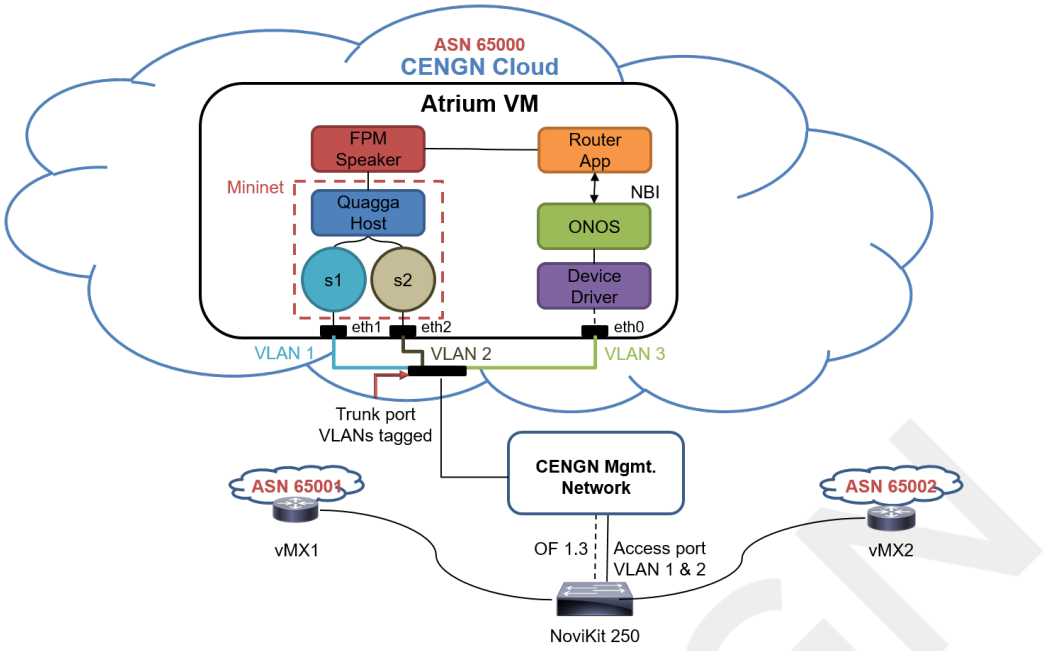

Fig. 2. TELUS-Noviflow POC network architecture

exchanging BGP routing information. Two Open Virtual Switch (OVS) $s1$ and $s2$ were also instantiated in Mininet that work as L2 bridges to connect the Quagga host to the Atrium VM interfaces. One interface of each switch connects to the Quagga host while the other is connected to the VM interface.

The interface of the compute node that is hosting the Atrium SDN stack is configured as a trunk port to carry the tagged VLAN traffic to the NoviKit 250 (VLAN 3) and to the peering BGP routers vMX1 (VLAN 1) and vMX2 (VLAN 2). The compute node (i.e., the CENGN cloud) is connected to the NoviKit 250 switch and the external BGP routers via a management network infrastructure. Different network segments of the testbed are discussed in the following subsections.

A. Atrium Router

The Atrium is an open source software-defined router that is able to speak different interior and exterior gateway routing protocols (Atrium Release 2016/A). In the testbed implementation at the CENGN cloud, the Atrium router consists of the ONOS-based Atrium distribution VM and a NoviFlow [7] software-defined switch named NoviKit 250. A schematic of the router is shown in Fig. 3. In this subsection, different components of the Atrium router implementation are briefly discussed.

The Atrium SDN stack consists of a Quagga router that implements the BGP protocol, an FPM speaker that communicates the BGP control plane information with the ONOS controller. The BGP routing information is forwarded to the Router App by the FPM

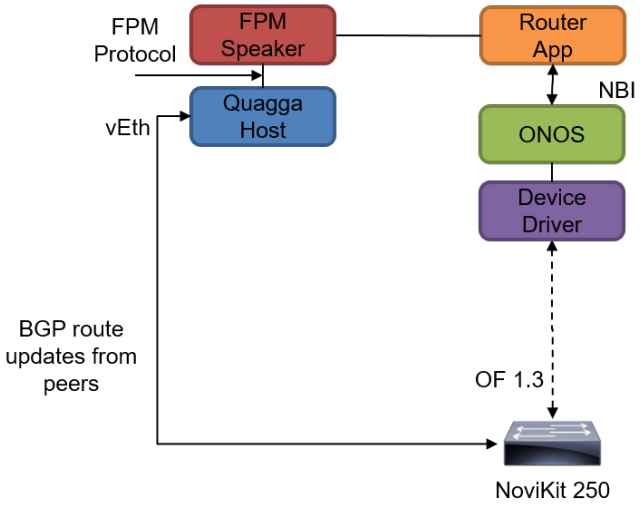

Fig. 3. The Atrium router architecture

speaker. The Router App translates the BGP information to the ONOS controller core using the north bound interface (NBI) APIs. In turn, the ONOS controller programs the data-plane (i.e., the NoviKit 250) flow-tables by the appropriate switch driver. The entries in the flow-tables are the flow rules (instructions) for the packets that traverse through the data-plane switch. Packets are treated (e.g., forward, drop, route to another flow-table, etc.) according to these flow rules. The ONOS controller installs all the proactive flows in Table-0 and the network derived flows into Table-1 of the NoviKit 250 switch.

As updated in the Atrium 2016/A release, the BGP peering traffic is redirected to the Quagga host in the data plane (cf. Fig. 2). This is done via a port (vEth) reserved for communication with the Quagga host. The BGP daemon in the Quagga host calculates the best routes from the received BGP routes from its neighbors. The Quagga host then uses FIB Push Interface (FPI) to push these best routes to a server named FPM.

The FPM server (FPM speaker) that is created in the ONOS controller core receives the best routes from the Quagga. The Router App uses the NBI APIs of ONOS, i.e., network graph, flow objective and intent to program the underlying switching fabric via ONOS controller core. The network graph is a directed, cyclic graph that comprises of network devices, links and end-station hosts. The intent helps to create a network-centric abstraction for programming data plane devices in a topology independent manner. And the flow objective provides a device-centric abstraction for programming data plane flows in a table pipeline independent manner. It allows applications to seamlessly work across different types of OpenFlow data plane pipelines.

ONOS provides a rich southbound abstraction where different network elements, such as switches, hosts and links are abstracted as a generic object. This in fact, allows the ONOS

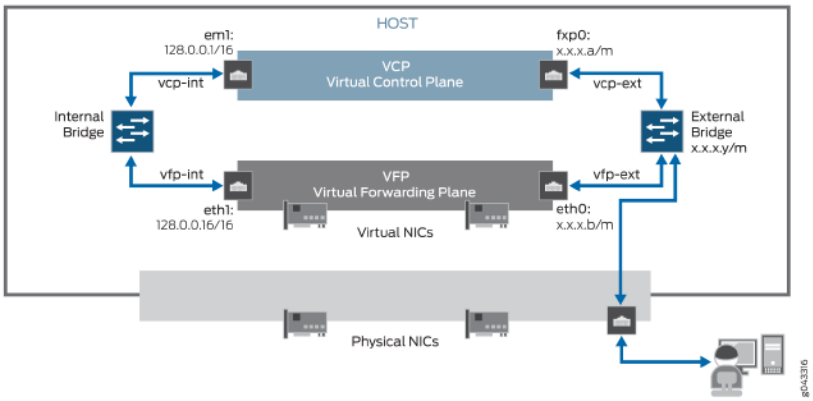

Fig. 4. Architecture of a vMX router

core to be agnostic of southbound protocols and forwarding devices. For the Atrium SDN stack implemented in CENGN cloud, the OpenFlow protocol specifics are abstracted by the driver layer plug-in. The ONOS controller programs the forwarding table of the NoviKit 250 switch via this driver layer.

B. Virtual MX (vMX) Router

The vMX router [5] is a virtual implementation of Junipers MX Series 3D Universal Edge Router. It runs Junos operating system (OS) and has the same configuration & management like the physical MX Series routers. The vMX routers used in this POC testbed were implemented in standard x86 servers running Linux operating system. Installation of the router was done by running an orchestration script that was included with the software package.

The use of vMX has the advantage of lower capital expenditure (CAPEX) and operational expenditure (OPEX) for network deployment as it is implemented in bare metal server rather than special purpose hardware platform. It also facilitates rapid introduction of new services, customized & personalized service provisioning to customers and dynamically scaling network operations as demanded by the network. The vMX is built around a three-layer architecture: the vMX router instance in the application layer; the Linux host OS, third party software, the KVM hypervisor in the virtualization layer and the x86 server as the physical layer. Fig. 4 shows the architecture of a vMX router instance.

The vMX router instance contains two VMs: one for the Virtual Forwarding Plane (VFP) and the other for the Virtual Control Plane (VCP). The VFP VM runs the virtual Trio forwarding plane software for packet forwarding and the VCP VM runs Junos OS as the control plane OS. The hypervisor virtualizes the physical network interface card (NIC) and presents it to the VFP VM as a virtual NIC. The Junos OS Command Line Interface (CLI) is used to configure the vMX interfaces in the VCP. Three Ethernet types are supported in vMX: Gigabit Ethernet (GE), 10-Gigabit Ethernet (XE) and 100-Gigabit Ethernet (ET).

Layer-2 connectivity is established between the VCP and VFP VMs via an internal bridge and between vMX VMs and management port via an external Bridge.

3. Deployment Challenges & Solutions

A number of challenges were encountered during the POC testbed deployment. This section briefly discusses some of these problems and the measures devised to resolve them.

A. Integration of the Atrium router in the CENGN cloud

In this PoC, the Atrium router is instantiated in CENGNs OpenStack cloud platform which is different from the VM deployment of the router demonstrated in the Atrium project [2]. Three different VLANs were presented to the Atrium distribution VM for isolated transport of BGP traffic from two peering routers (vMX1 and vMX2) and for transporting the control channel traffic from the ONOS controller to the NoviKit 250 switch. In order to route BGP traffic in the data plane (through the NoviKit 250 switch) from the peering routers vMX1 and vMX2, two virtual switches $s1$ and $s2$ were instantiated in Mininet [6] in two separate VLANs, 1 and 2, respectively. And the control channel information from the Atrium router to the NoviKit 250 switch was transported through the VLAN 3. A trunk port was used to carry the tagged traffic of the VLANs.

B. Switch Driver Modification

The software-defined switch used as the forwarding plane of the Atrium router was NoviKit 250 from one of the project partners NoviFlow [7]. The default Atrium router driver requires some configuration to enable the control of the NoviKit 250. In the driver configuration, the default hardware version was changed to NoviKit 250 before ONOS controller could be recompiled to allow for smooth communication with the switch.

C. Allowing Atrium VM Traffic Through OpenStack

When the Atrium SDN stack is instantiated in the CENGN cloud, the traffic from the VM could not pass through the cloud infrastructure. The reason is, by default OpenStack [8], as part of its security features, sets iptables rules in the Linux bridge in the computing nodes to allow traffic from certain IP and MAC address pairs. As the Quagga host interface address was not recognized in the iptables, the packets from the Atrium VM was dropped. Manually installing the iptable rules does not provide a permanent solution to this problem. To resolve this issue, the *allowed address pair* extension of Neutron [8] was used to to allow additional IP/MAC address pairs from the Quagga host interfaces. The *allowed address pair* extension extends the port attribute to enable a network administrator to specify arbitrary MAC/IP pairs that are allowed to pass through a port regardless of the associated subnet with the network. This automatically populated the iptables of the Linux bridge with the correct IP/MAC address pair and thus traffic from the Atrium VM could pass through the OVS, so that, the BGP neighbors (i.e., vMX1 and vMX2) could communicate with the Atrium router.

D. VLAN Separation Issues

This POC demonstrates Atrium router peering with two different routers (peers) in two different ASes. For this reason, segregation of control channel information from the Atrium VM to the NoviKit 250 was necessary. This was accomplished by using two separate VLANs, one for ASN-65001 and the other for ASN-65002, so that, the control channels remain separated while using the same switch port.

E. Switch Physical Link Issue

The NoviKit 250 switch used in this POC demonstration had some physical link compatibility issues working with Juniper EX2200 switch that is part of CENGNs secured switch infrastructure. While the built-in copper management ports in the NoviKit 250 worked fine with the EX2200 switch, the Small Form-factor Pluggable (SFP) based control plane port had problem and an *Alignment Rx error* was thrown. The possible reason is mismatch in interface speed or in duplex mode operation with the SFP transceiver module used with the NoviKit. As a quick work-around for this issue, the NoviKit control port was connected to the Atrium VM via an EdgeCore ECS4510-52T switch, which did not experience the same physical link issues. The switch vendor, NoviFlow which is one of the partners of the project have been informed about it and have identified a possible SFP transceiver incompatibility with the EX2200s copper ports. NoviFlow is investigating alternate SFP transceivers to resolve the issue.

4. Conclusion

Current vibrant Next Generation Networks (NGN) ecosystem consists of a significant number of open source software in SDN and cloud computing technologies. For building future carrier-grade network solutions it is imperative that these open source software packages be robust and they integrate in a cohesive manner. Since different software are built independently, various compatibility issues are raised while integrating them together. The incompatibility or lack of interoperability stems from the lack of standardized interfaces between different software packages. This project addresses this particular issue and demonstrates the challenges associated in integrating different open source software to build an SDN stack that provides end-to-end network solution. The project aim was the integration of Atrium router within OpenStack cloud infrastructure. And in doing so, integration was done is different layers: underlay integration of OpenFlow-based NoviKit 250 switch with traditional switches; SDN overlay integration of the Atrium software stack in OpenStack cloud platform; integration of ONOS controller with NoviKit 250 switch and integration of software-defined Atrium router with traditional Juniper vMX routers.

During the implementation, it was evident that it is not a straightforward task to take different open source software and integrate them together to build a network solution. A lot of challenges needed to be resolved which are discussed in Section 3. But in the end, we

were able to successfully integrate the SDN-based Atrium router in CENGNs OpenStack-base cloud infrastructure. And we successfully deployed a BGP network consisting of both software-defined router and traditional routers that connected multiple ASes. We believe that for a sustainable growth of NGN (composing of different open source software) it is important to make the deployment experience public, so that, the community can learn (from each others experiences) and collaborate together.

This POC demonstration provides the project partner, TELUS, a practical use case of SDN controller in its Converged Edge Network. TELUS has adopted whitebox strategy for its customer premises equipment (CPE) line up, and it might be interesting for TELUS to expand this strategy towards its edge network. The widely used protocol at the edge of networks is multi-protocol label switching (MPLS). Though some SDN controllers can now support MPLS encapsulation at data plane with OpenFlow as control plane, most likely software defined MPLS network will start with brown-field integration, which will require certain signalling between SDN controller and existing MPLS network for MPLS label distribution, VPN route exchange and traffic engineering etc.

There are a few important lessons that the project partners were able to learn from this project:

- the integration of Atrium router in cloud environment gives critical insight to software-defined network function virtualization (SD-NFV) deployment;
- requirement for commercialization of Atrium-like software-defined routers;
- possible learning curve for the engineering team in adopting such open source-based solutions;
- deployment challenges and guidelines for operators who plan to adopt such cutting edge software-defined cloud-based networking solutions;
- The POC provided TELUS a practical use case for SDN controller in its Converged Edge Network.

Acronyms

ASN Autonomous System Number.

BGP Border Gateway Protocol.

CLI Command Line Interface.

FIB Forwarding Information Base.
FPI FIB Push Interface.
FPM Forwarding Plane Manager.

NGN Next Generation Networks.

ONF Open Networking Foundation.
ONOS Open Networking Operating System.
OVS Open Virtual Switch.

POC Proof of Concept.

SDN Software Defined Networking.
SFP Small Form-factor Pluggable.
SME Small and Medium Enterprise.

VCP Virtual Control Plane.
VFP Virtual Forwarding Plane.
VLAN Virtual Local Area Network.

References

[1] ONF, *Open Networking Foundation*, available at https://www.opennetworking.org/ (accessed on June 26, 2016).
[2] Atrium, available at https://github.com/onfsdn/atrium-docs/wiki (accessed on June 26, 2016).
[3] ONOS, *Open Networking Operating System*, available at http://onosproject.org/(accessed on June 26, 2016).
[4] Quagga, *Quagga Routing Suite*, available at http://www.nongnu.org/quagga/ (accessed on June 26, 2016).
[5] vMX, *Getting Started Guide*, Release 15.1F3, Juniper Networks Inc., Oct. 2015.
[6] Mininet, *Mininet: An Instant Virtual Network on your Laptop (or other PC)*, available at http://mininet.org/ (accessed on June 26, 2016).
[7] Noviflow, available at http://noviflow.com/ (accessed on June 26, 2016).
[8] OpenStack, *Open Source Software for Creating private and Public Clouds*, available at https://www.openstack.org/ (accessed on June 26, 2016).