Virtual lab for SDN and NFV - concept and architecture

Juraj Londák[1], Martin Medvecký[1], Tomáš Tóth[1], Pavol Podhradský[1]
[1] Slovak University of Technology in Bratislava, Ilkovičova 3, 812 19, Bratislava, Slovakia
martin.medvecky@stuba.sk

Abstract –**In the paper, the concept and architecture of SDN and NFV virtual lab are presented. This lab enables the installation, configuration and testing of selected SDN and NFV networks for both research and educational activities.**

Keywords – **SDN, NFV**

I. INTRODUCTION

The SDN (Software Defined Networking) define the concept of the physical separation of the network control plane from the transport plane where the control plane is operated by several devices. SDN is probably one of the most significant shifts in network industry, which has been seen in recent years. SDN is being developed because of the high demand for keeping up with new networking requirements, the emerging trend of remote data management, cloud computing, mobile data processing and processing of big data.

A. *SDN Architecture*

SDN architecture has the following features:

- Directly programmable - Network control is separated from transport functions, so direct programming of network control is possible,
- Agile - Administrator allows dynamic fluctuation of traffic throughout the network to meet ever-changing customer needs,
- Centrally controlled - Network Intelligence is centralized in a software controller to provide global oversight over the entire network that appears to applications as a logic switch,
- Programmable - SDN allows network administrators to configure, manage, and optimize the network very quickly through dynamic, automated SDN programs that can be written by themselves,
- Based on open standards and devices from different vendors - New implementations are implemented through open standards, simplifying the construction and performance of the network because commands are provided by the SDN controller instead of multiple specific devices from different vendors.

Unlike traditional network architecture, SDN allows applications to gain more information about network status through the controller. The following three components are involved (see Fig.1):

- SDN Applications - Programs that communicate via an API with an SDN controller. In addition, applications can collect information from the controller for decision-making purposes.
- SDN controller - a logical unit that receives instructions from the application layer and transfers it to network elements. This communication also works in the opposite direction, i.e., the controller extracts data from hardware network devices and transfers them back to applications.
- SDN network devices - provide network data transmission.

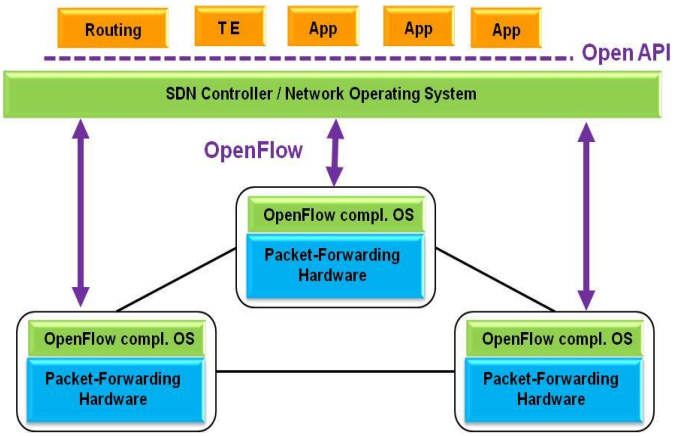

Figure 1. SDN architecture

The SDN functionality is spread over several layers:

The forwarding plane - is responsible for handling the packets on the basis of instructions received from the control plane. Plane functionality includes forwarding, discarding, and modifying packets. This layer captures incoming packets, performs their primary inspection (e.g., reports unrecoverable errors), and determines their subsequent processing by the network. In the received packet, the recipient's address is identified and the packet is forwarded according the information stored in the Forwarding Information Base (FIB) that was previously created by the control layer. This method is called a "fast way" or a fast forwarding when for the correct forwarding

of the packet, it is enough to look only to the FIB. An exception occurs only when the destination can not be determined by the FIB because it does not contain the corresponding record; in that case the information about the packet is sent to the controller.

B. *NFV Architecture*

The architecture of NFV technology was designed by Network Functions Virtualization Industry Specification Group (ETSI NFV ISG) based on the following components:

- NFVI (NFV Infrastructure) - provides virtual resources needed to support the implementation of virtualized network functions,
- VNF (Virtualized Network Functions) - software implementation of network functions that is able to run by NFVI,
- NFV MANO (Management and Orchestration) - covers orchestration and lifecycle management of physical and / or software tools that support the virtualization and infrastructure lifecycle management.

NFVIaaS (Network Functions Virtualisation Infrastructure as a Service) - is a service providing an environment in that a various network virtual devices and functions can be provided. This environment is referred to as "Virtualized Network Functions" (VNFs). NFVI can be compared to IaaS (Infrastructure as a Service) by its very nature and, since it supports networking on demand, it is comparable to NaaS (Network as a Service). IaaS and NaaS are functionally components of the NFVI infrastructure and are mapped as is shown in Fig.2.

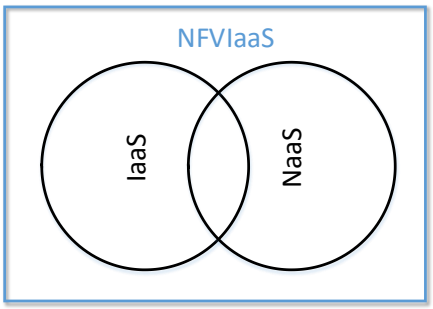

Figure 2. Mapping of IaaS and NaaS within NFV infrastructure

The resources that are associated with these services (IaaS, NaaS) are physical resources, storage resources and computational resources. In the NFV model, these resources are considered as computational, hypervisor and network domain of NFVI. In general, NFV infrastructure computational nodes are located at NFVI-PoP points i.e. Central Office (CO), outside the CO in external branches, in dedicated outlets or directly integrated into other network elements or mobile devices. Physical location of infrastructure is largely irrelevant to Cloud Computing services, but many network services have a degree of dependence on location.

With resource pooling, it is possible that a single pool of resources provides multiple applications from different administrative domains and domains of confidentiality (referred to as multi-tenancy). The goal of virtualization for NFVI is to provide an environment in which different software entities are created. NFVIaaS should provide connectivity between the virtual network and the physical network. Infrastructure must be designed to uniquely separate the resources assigned to individual users and prevent the failure of one user from causing a failure to another user.

II. VIRTUAL LAB FRAMEWORK

The virtual SDN and NFV laboratory was designed for teaching and research purposes to enable study, verification, testing and examination of SDN and NFV technologies by a secure and controlled manner for both local and remote users.

For this reason the emphasis was put on:

- Openness
- Scalability
- Efficiency
- Safety
- Manageability

For this reason, we have focused on technologies and solutions that make it possible to meet these requirements. The virtual laboratory will be based on the following components (see Fig.3):

- Open Stack
- OpenDalilight
- OPNFV

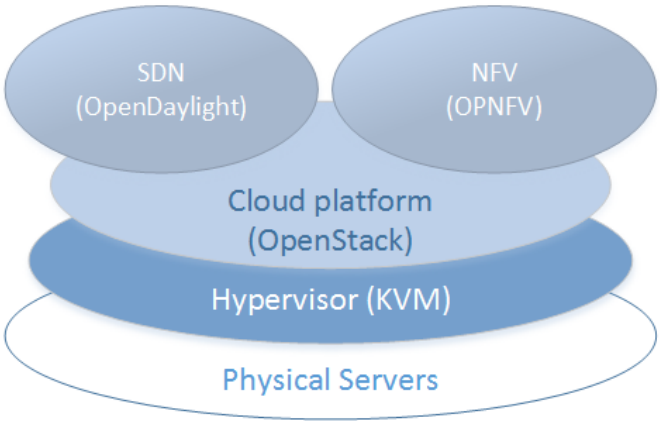

Figure 3. General architecture of SDN and NFV virtual lab

A. *OpenStack*

OpenStack is a cloud-based software platform that provides simple implementation, high-availability and support for a large number of features. It does not have any proprietary hardware or software requirements and is designed to work with both fully-virtual and bare-metal systems. It also supports several hypervisors such as: KMV, XenServer, VMware, and container technologies such as LXC. OpenStack therefore has a wide range of applications from proprietary service providers to large corporate IT departments that use it to create private clouds for their own needs. Since 2014, OpenStack has been backed by VMware and has been fully integrated into their infrastructure, allowing OpenStack to easily control the VMware cluster.

The main parts of OpenStack architecture are (see Fig. 4):

- *Cider* – a block storage system
- *Glance* - provides discovery, registration, and delivery services for disk and server images
- *Heat* – provides orchestration - automated instances preparation
- *Horizon* – a dashboard with a graphical interface allowing to access, provision, and automate deployment of cloud-based resources
- *Keystone* - a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system in our virtual SDN and NFV laboratory
- *Neutron* - a system for managing networks and IP addresses
- *Nova* - a cloud computing fabric controller
- *Swift* - a scalable redundant storage system

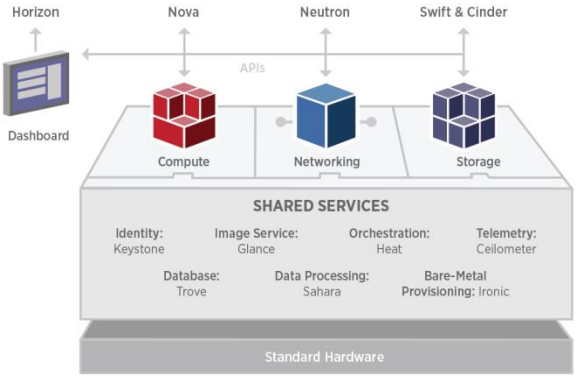

Figure 4. Open Stack

The virtual laboratory architecture is shown in Fig. 5. The proposal has an emphasis on high availability and includes device redundancy. Device redundancy makes it possible for a backed up device to be replaced by another secondary device in the event of a device failure. Redundancy is maintained for all devices, except for data storage, where redundancy is only addressed within the disks contained in the repository. As a result, in the case of hardware or software malfunctions, the data storage device is not backed up as well.

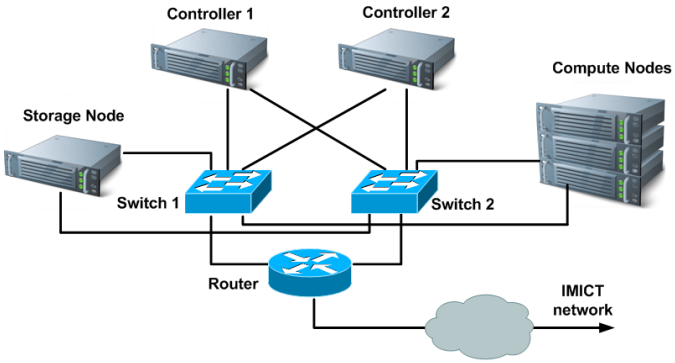

Figure 5. Virtual lab architecture

Redundant are only disk arrays and data stored on them. In the case of a power failure, the power supply will be secured from the backup source by a UPS.

B. *OpenDaylight*

The main part of our SDN virtual laboratory will be based on the OpenDaylight platform. The OpenDaylight is a modular open source SDN platform developed to promote SDN and NFV. The software is written in the Java programming language. OpenDaylight enables network services across a spectrum of hardware in multivendor environments. The implemented microservices architecture allows users to control applications, protocols and plugins, as well as to provide connections between external consumers and providers.

OpenDaylight employs a model-driven approach to describe the network, the functions to be performed on it and the resulting state or status achieved. In the OpenDaylight Model Driven Service Abstraction Layer (MD-SAL), any application or function can be bundled into a service that is then loaded into the controller. Services can be configured and chained together in any number of ways to match fluctuating needs within the network. This opens a wide range of options for virtual lab users to test different SDN settings and configurations.

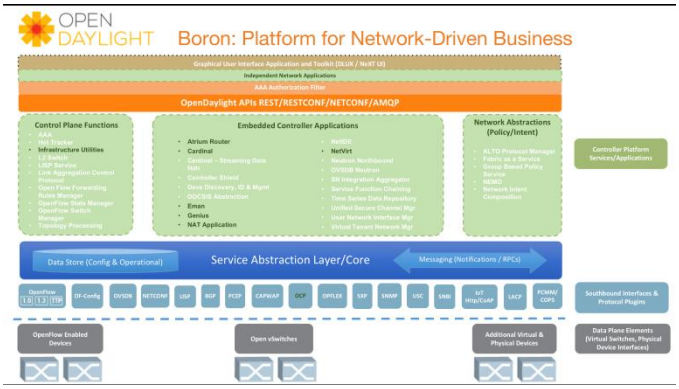

Figure 6. OpenDaylight architecture

C. *Open vSwitch*

Open vSwitch (OVS) is a virtual multi-layer network switch that enables efficient network automation while supporting standard management interfaces and protocols such as NetFlow, sFLOW, SPAN, and so on.

The main purpose of the Open vSwitch is to provide a switching stack for virtualization of hardware environment and support of multiple protocols and standards used in computer networks. Open vSwitch includes a database server (ovsdb-server), vSwitch deamon (ovs-vswitchd) and possibly a forwarder module. The control and management part consists of controllers and managers. Managers use the OVSDB management protocol for Open vSwitch management. OVS can support multiple logical data paths, referred to as "bridges".

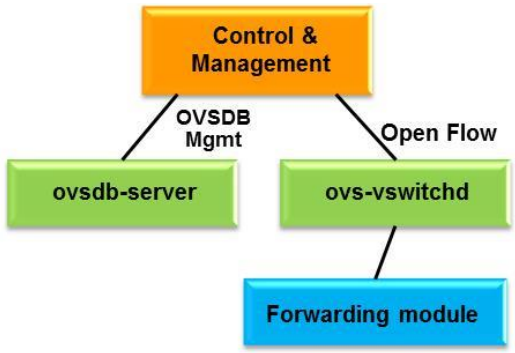

Figure 7. Open vSwitch architecture

The operations supported by OVSDP include:

- Creating, modifying and deleting OpenFlow data paths (bridges)
- Configuring the OpenFlow Controller Group
- Configuring the OVSDB manager group
- Creating, modifying and deleting OpenFlow ports
- Creating, modifying and deleting the OpenFlow interface
- Configuring QoS
- Collection of statistical data

D. OPNFV

Within the Virtual Lab, the NFV functionality will be realized by the OPNFV (Open Platform for NFV) [4]. As a testing and integration project, OPNFV brings together upstream components across compute, storage and network virtualization in order create an end-to-end platform. The present OPNFV release, Danube, builds and integrates multiple end-to-end networking stacks, including MANO, data plane acceleration, and architecture advancements.

III. CONCLUSIONS

This paper presents the concept and general architecture of the SDN and NFV virtual lab. The architecture is based on OpenStack software platform. SDN segment of the architecture is build up on Opendaylight platform and NFV segment on the OPNFV (Open Platform for NFV).

ACKNOWLEDGMENT

Research described in the paper was financially supported by the H2020 project NEWTON, No. 688503 and VEGA project INOMET, No. 1/0800/16

REFERENCES

[1] Open vSwitch 2.5.0 Documentation, http://openvswitch.org/support/dist-docs-2.5/

[2] OpenStack, Documentation for Ocata (February 2017), https://docs.openstack.org

[3] OpenDaylight Documentaion, https://www.opendaylight.org/start

[4] OpenNFV Documentation, https://www.opnfv.org/resources

[5] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (references)

[6] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[7] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.