VIRTUALIZATION IN OPENFLOW NETWORKS

Miladin Tomić[*], Milutin Radonjić [**], Neđeljko Lekić[**], Igor Radusinović[**]

***Keywords**: FlowVisor, OpenFlow, SDN, virtualization.*

Abstract: Network virtualization allows network resources to be shared, by dividing them and creating individual networks, which can operate independently of each other. This paper describes FlowVisor, as a tool for network virtualization. FlowVisor is a special purpose controller, which enables virtualization in OpenFlow networks, by dividing a physical network into multiple logical networks. It ensures that each virtual network uses only its own resources. The paper provides brief overview of the OpenFlow technology, and FlowVisor's role in it. Furthermore, it is given a short insight into FlowVisor basic work principles and main goals which it has to accomplish, as well as the explanation how it provides virtualization of the network resources. We have implemented FlowVisor in our OpenFlow testbed, and demonstrated its ability to provide coexistence of multiple virtual networks in one physical network.

1. INTRODUCTION

Although network virtualization is attractive for research community for a long period of time, it is in the focus since Software Defined Network (SDN) concept has appeared. The separation of the control plane and data plane, through SDN, has opened many opportunities in networking research, particularly in the field of network virtualization [1].

Before SDN concept has appeared, full virtualization could be achieved only by the dedicated hardware, often with duplicated components. This approach is impractical and very expensive [2]. Creation of two virtual networks imposes the need for the devices with

[*] Miladin Tomić, spec. sci, Faculty of Electrical Engineering, University of Montenegro, Podgorica (e-mail: miladin.toomic@gmail.com)

[**] Doc. dr Milutin Radonjić, Prof. dr Neđeljko Lekić, Prof. dr Igor Radusinović Faculty of Electrical Engineering, University of Montenegro, Podgorica (phone: +382-20-245873; e-mail: {mico, nedjo, igorr}@ac.me)

doubled components. For larger number of virtual networks it is questionable is it possible to manufacture such devices at reasonable price, and who would buy it, keeping in mind inflexibility of network in which they would be deployed. Significant improvements in the commercial networks virtualization could not be achieved, because vendors are usually not willing to reveal the internal structure of their devices.

OpenFlow, as the most promising SDN technology, solves the problem related to network virtualization to some extent [3]. Therefore, OpenFlow is envisioned as the technology that will speed up the development of new network management solutions. In that sense, it is necessary to provide realistic topologies to test proposed solutions. Building completely new networks for that purpose is not cost effective. Therefore, the idea is to virtualize commercial networks and thus provide realistic topologies for testing. This can be performed by the OpenFlow special purpose tool, called FlowVisor.

It would be suitable to make comparison with some commercial technologies in order to get insight into radical new approach that FlowVisor brings to the network virtualization. Commercial technologies, even with prefix virtual in their names, essentially do not provide the full virtualization. One example is a concept of Virtual Local Area Network (VLAN), which is technology for segmentation of single broadcast domain into several independent broadcast domains (e.g. a primitive traffic separation on link layer, based on switches interfaces). Virtual Private Network (VPN), as solutions used primarily across Wide Area Networks (WAN), is technology for enabling secure communication between end points, so it cannot be considered as a virtualization tool. On the other side, FlowVisor enables virtualization of all network elements, in very flexible manner, and thus provides coexistence of multiple virtual networks, with guaranteed resources, on top of the single physical network.

OpenFlow's possibility to create new network management solutions has already been demonstrated in earlier research [4]. Considering great potential of network's virtualization, not only to provide realistic testbeds, but to provide base for new network concepts (for example, networks as a service), further research activities in SDN were focused in that direction. We had deployed OpenFlow testbed, and created multiple virtual networks by usage of FlowVisor within it. Thus, we demonstrated FlowVisor's ability to virtualize network resources. In this paper we will provide insight into FlowVisor's concept and mechanisms used to virtualize all network resources, as well as the OpenFlow testbed that we have successfully virtualized.

Paper is organized as follows: Section 2 gives short insight into OpenFlow technology. In Section 3, FlowVisor is presented, and parallel with the virtualization principles in the computer world is given. Section 4 lists major demands that FlowVisor must satisfy, and which are base for its development. FlowVisor architecture is described in Section 5. Section 6 explains mechanisms for virtualization of network resources. Final conclusions are given in section 7.

2. OPENFLOW TECHNOLOGY

The major change that OpenFlow introduces into networks is the separation of the control and data plane. That architectural change enables radical new approaches in network control. In OpenFlow paradigm, network is clearly separated into two parts: data

plane, which consists of devices without intelligence, and control plane, which has all network intelligence, and manages the data plane.

Intelligence is physically located at the single place, called controller. The controller is the device that runs control application for managing devices in the data plane. Devices in the data plane are just store–and-forward devices that direct traffic through the network. This concept is illustrated in Figure 1.

In non-OpenFlow devices (e.g. switch), forwarding decisions are made locally (within device), by looking in the appropriate table (MAC table). In accordance with the records in the table, frames are forwarded to the appropriate interfaces. If requested record does not exist in the table, frame will be broadcasted. All decisions are made within devices, and intelligence is implemented in their control plane. This network control approach is known as decentralized control. On the other hand, in OpenFlow network all devices are controlled by a single controller, and this approach is known as centralized control. The centralized control imposes some issues regarding scalability and overload at the controller, which are addressed in [5].

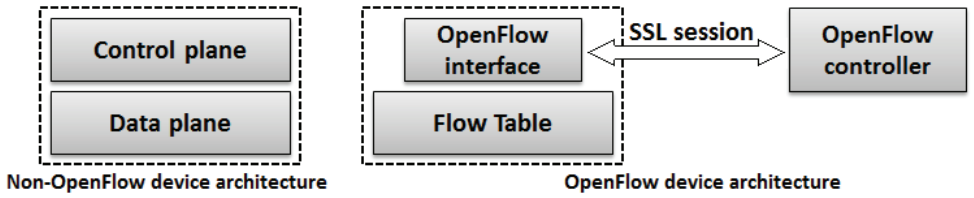

Fig. 1 Comparison between different architectures in non-OpenFlow devices and OpenFlow devices

Conceptually, OpenFlow switch operates similarly to the non-OpenFlow switch. But, if appropriate record does not exist in the switch table, frame will not be broadcasted, but sent to the controller through dedicated OpenFlow interface. That interface is predefined for connection with the controller. Before sending the frame, it will be encapsulated in appropriate OpenFlow message. Format of the message and overall communication between controller and switch relays on OpenFlow protocol [6].

After reception of the message, control application makes a decision about traffic handling. Decision is formatted into appropriate set of instructions, and then sent back to the switch. After reception of this message, switch will install set of instructions into the flow table. The purpose of the flow table is the same as MAC table in non-OpenFlow switch: to determine traffic forwarding, based on its records. However, there are some huge differences between non-OpenFlow and OpenFlow switches. While non-OpenFlow switch makes decisions based only on MAC addresses, OpenFlow switch makes decisions based on 10 parameters: from physical interface to the transport layer port numbers. Parameters used by OpenFlow switch to direct packets are shown in Figure 2.

Because OpenFlow switch does not forward traffic based only on MAC address, it is not appropriate to say that it forwards frames. Instead, it is more appropriate to say that it directs flows through the network. Flow is a group of packets that find match in one of the records in the flow table. Flow tables represent abstractions of switches, and flows are

abstraction of traffic. As the records in flow table can be defined in flexible manner, the flows can also be defined in various ways.

	Link layer				Network layer			Transport layer	
Interface	Source MAC address	Destination MAC address	Ethernet Type	VLAN ID	Source IP address	Destination IP address	IP protocol	Source port number	Destination port number

Fig. 2 Header fields used for flows definition

So, the controller manages the data plane by populating the flow table, and thus controls direction of the flows through the network. The controller can populate, delete or modify the flow table records, and ensure some statistical data related to switches.

The separation of the control and data plane, abstraction of data plane with flow tables, and control from the centralized location is an approach that provides ability for easy creation of new network management solutions. It also offers the opportunity for full network virtualization, as it will be explained in the following sections.

3. FLOWVISOR WORKING PRINCIPLES

For better understanding of principles on which FlowVisor is built, it is suitable to make comparison with virtualization in the computers world. In the computers world, instruction set provides abstraction of hardware resources. Above that, some tools for virtualization (e.g. VMware, or Qemu) conduct slicing of abstracted hardware resources (memory, processor, etc.). At the top of the sliced abstracted resources, different operating systems can operate in parallel. These separated and isolated resources are sometimes referred to as virtual machines. This is shown in Figure 3.

Computer virtualization gained in popularity, and most of today's servers are virtualized in order to reduce costs and increase their utilization. FlowVisor was developed following that successful concept. Having OpenFlow as a tool for abstraction, FlowVisor is placed between OpenFlow switches (which are represented as flow tables) and controllers. Thus, it separates and isolates abstracted hardware, in order to offer isolated parts of infrastructure to the superior controllers. To provide network virtualization, it is necessary to divide and isolate following resources: bandwidth, flow tables, Central Processing Unit (CPU) of devices, topology and traffic. Virtualization of these resources means that every logical network provided by FlowVisor will have certain amount of each of those resources. Virtualization, e.g. dividing and isolation of previously mentioned resources is called slicing, and single slice represents precisely defined amount of each of those.

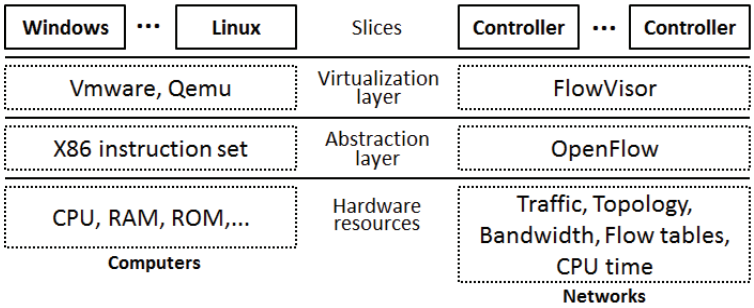

Fig. 3 Comparison of virtualization principles in computer's world and network's world

4. DEMANDS THAT FLOWVISOR HAS TO SATISFY

While FlowVisor was being designed, it was very important to ensure ability of flexible defining of virtual networks, transparency towards the control plane and the data plane and strong isolation among the virtualized resources. As the primary goal of network virtualization is to create environment for testing a new ideas, it is important that resources allocation policy be flexible as much as possible. Provided amount of flexibility determines the range of virtual environment that can be created, and thus, the range of ideas that can be tested.

The transparency demand is imposed by need to ensure dynamical development of new control solutions. If this demand is not satisfied, both network hardware and control solutions must be aware of the virtualization layer presence. That would be the limit from perspective of easy development of new solutions, because new solutions would be designed taking into account FlowVisor's presence. Thus, a new ideas could not be developed and tested in the realistic environment, but instead, they would be designed to adapt to the virtualization layer. Hence, it is of crucial importance to ensure that virtualization layer will be transparent towards both the control plane and the data plane, in order to simplify and boost the development of new network management solutions.

The last mandatory demand is the demand for isolation. Creation of multiple virtual networks is not achievement by itself. If one virtual network can make impact on resources of some other virtual network or networks, there is no virtualization. We can talk about virtualization only in case when virtual networks can make impact on resources that are delegated to them by appropriate resource slicing policy. So, ensuring strong isolation among isolated resources is necessary, in order to provide coexistence of multiple virtual networks at the top of one physical network [7].

5. FLOWVISOR ARCHITECTURE

FlowVisor has three functional entities: module for resource allocation, module for translation, and module for forwarding, as it is shown in Figure 4. These modules are engaged in different phases of network's virtualization process.

Because the FlowVisor is hierarchically located above the OpenFlow switches, it monitors their flow tables and flows. By monitoring these two resources, FlowVisor monitors whole data plane of the network. These resources, with bandwidth, topology and CPU of devices in the data plane are divided and isolated (e.g. sliced). Those sliced resources are then controlled by different controllers. By slicing network resources and then delegating slices to the appropriate controllers, FlowVisor provides network virtualization [8].

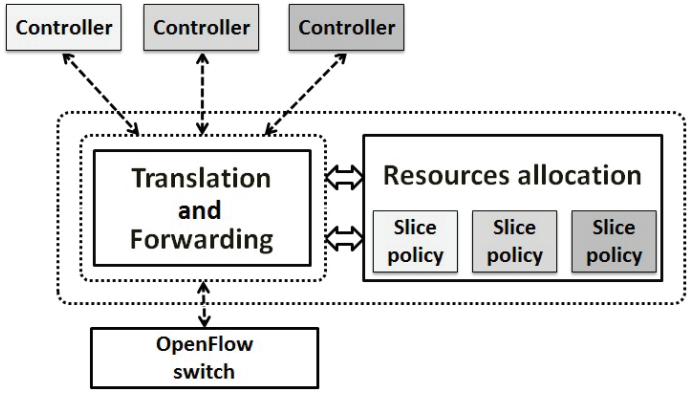

Fig. 4 FlowVisor's architecture and its location in the OpenFlow network

As it was mentioned earlier, all OpenFlow messages from switches, and all control messages from controller are delivered through the SSL session, in accordance with the OpenFlow protocol. However, by locating FlowVisor between OpenFlow switches and controllers these sessions are transformed into two independent sessions: first session from the switches to the FlowVisor and second session from the FlowVisor to the controllers (thanks to FlowVisor, more than one controller can exist within one physical network). The described session's transformation is invisible for the switches and controllers. This structure is illustrated in Figure 5. All OpenFlow messages generated both from the data plane and the control plane will be intercepted by the FlowVisor.

Following SDN concept, even the FlowVisor is just an application. It is composed from previously mentioned three logical modules, which are intended to provide network virtualization (Figure 4). The first of them is module for translation. It is enrolled every time when messages arrive both from the switches and the controllers. It examines the messages and determines from which devices they have arrived, and where they are addressed to, e.g. on which resources they are trying to make an impact. Then, it communicates with the module for resources allocation (the second module). This module is used to check whether message violate appropriate slicing policies, and if whether it may be corrected or must be dropped. In order to resolve it, the resource allocation module consults slice policies. These policies describe virtual networks and hold information about resources delegated to each. After resolving, appropriate instruction is sent to the forwarding module. This module will forward message to a corresponding device in the control or data plane, or drop the message in case it violates slice policy.

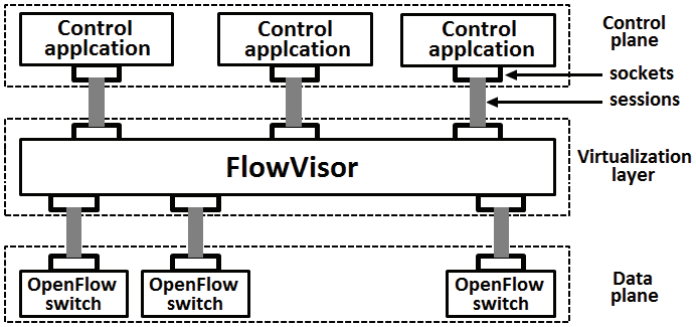

Fig. 5 OpenFlow network with FlowVisor, seen from TCP/IP Layer 4/5 perspective

By using these three modules, it is ensured that every controller obtains only messages and resources, that has rights to. FlowVisor behaves as a transparent OpenFlow proxy, which communicates with control and data plane using OpenFlow protocol. By interception of messages from both sides, and rewriting or dropping them, it can provide coexistence of multiple virtual networks on one physical network.

6. MECHANISMS FOR RESOURCE VIRTUALIZATION

Since FlowVisor virtualizes five resources, virtualization process can be divided into five sub processes, which is virtualization of: traffic, topology, CPU of devices, flow tables and bandwidth.

Because the traffic is represented by flows, overall traffic is divided by delegating different groups of flows to different slices. Group of flows delegated to a certain slice is known as FlowSpace. FlowSpaces are defined as set of records, where each record is consisted of rules and actions. Rules define flows, while actions define what will be done with flows. Possible actions are: *allow*, *deny* and *read-only*. Action *allow* gives full control over the defined flow, *deny* means its rejection, while *read-only* means that no control actions can be conducted over the flow, but the controller can use the traffic for monitoring purposes.

To get clear insight on how FlowVisor divides and isolates network resources, one of the examples that we implemented will be considered. OpenFlow network has been virtualized and two virtual networks have been created, as it is shown in Figure 6. Whole HTTP traffic from the source IP address is delegated to the first virtual network, while all other traffic is delegated to the second virtual network. Traffic division is performed by the first virtual network FlowSpace definition as:

$$\text{Allow: TCP\_port} = 80 \ \text{IP} = \text{address\_1} \tag{1}$$

The virtual network's controller which has this record in FlowSpace's definition has full control over HTTP traffic (assuming that only HTTP uses port 80, which was the case in our example) from IP address_1. Since the controller itself is not aware of the FlowSpaces existence, it thinks that controls all the traffic. Therefore, it is possible that controller tries

to make an impact on the traffic that is not delegated to him by appropriate slicing policy. Such an attempt will be intercepted by the FlowVisor. In this particular case, the message that holds instructions to make an impact on non HTTP traffic from the IP address_1 will be discarded. Thus, FlowVisor ensures that every controller can control only the traffic that is delegated to it, in the process of traffic virtualization.

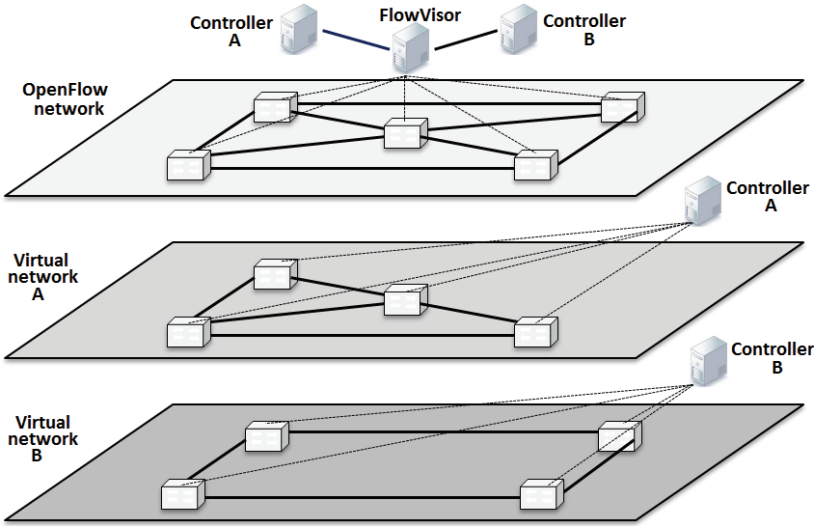

Fig. 6 OpenFlow network that has been virtualized

Besides keeping the track on messages that arrive from the controller, FlowVisor also intercepts the messages that are sent from the switches. Thus, it is guaranteed that OpenFlow messages generated in data plane by the HTTP traffic from the IP address_1 will be forwarded to the appropriate controller.

On the other side, to ensure that traffic belonging to the first virtual network does not appear in the second virtual network, it is necessary to define FlowSpace of the second virtual network as:

$$\text{Deny: TCP\_port} = 80 \text{ IP} = \text{address\_1; Allow: all;} \quad (2)$$

This will result in the rejection of the HTTP traffic from the IP addres_1, and in the acceptance of all other traffic. Thus, FlowVisor slices traffic, while interception and modification (or rejection) of OpenFlow messages ensure traffic isolation. It means that the full virtualization of the traffic is achieved.

Beside the traffic, it is necessary to virtualize remaining four resources in order to ensure full network virtualization. Virtualization of the topology can be divided into two processes: virtualization of the data plane devices, and virtualization of the links among those devices. Devices will be detected when they try to connect to the controllers. FlowVisor will intercept that connections and ensure that every device can be connected only to the appropriate controller. One device can be shared among multiple controllers. In

that case FlowVisor will establish multiple sessions per controller. Links detection is accomplished by sending packets through every switch interface and receiving them on the other ends. Virtualization of links is conducted by prevention of controllers to send messages through the interfaces that are not delegated to them. For example, when the controller tries to send broadcast message without rights to all interfaces, broadcast will be transformed into multicast to the delegated interfaces, by appropriate slice policy.

It is also necessary to virtualize switch's processor, in order to prevent its overload from a single virtual network. Main source of switch's CPU load are the messages exchanged between the switch and the controller. If these messages are exchanged intensively, CPU resources could be overloaded. In order to avoid it, it is necessary to reduce the speed of messages for each slice. When speed increases above defined thresholds, messages are going to be discarded for a while. Hence, the CPU virtualization is conducted by limiting the speed of the OpenFlow messages exchanged between the switches and the controller of each slice. Mechanism for limiting these speeds should be implemented in the data plane devices and should be exposed to the FlowVisor. It is up to device manufacturer to provide this functionality [9].

Virtualization of the flow table is conducted by limiting the number of records for each slice. This is achieved by the counters for each slice. FlowVisor counts flow install messages, sent by each controller. When number of these messages, and thus records, overrides maximal allowed value, further recording will be disabled and controller will be notified that the flow table is full. Each record in the table has associated timer, which is defined by the flow record validity period specified by controller in the flow install message. Upon timer expiration, appropriate counter is going to be decreased.

FlowVisor by itself does not provide mechanism for bandwidth virtualization. Therefore, it has to be accomplished by the data plane devices. Bandwidth virtualization can be performed by mapping the flows of different slices (virtual networks) into different Quality of Service (QoS) classes. Mechanisms for providing QoS are significant filed of research, which permanently evolve [10], [11].

7. CONCLUSION

Development of the OpenFlow and consequently of the FlowVisor was guided by the idea to make a deflection from the currently ossified and highly inflexible network architecture and boost the development of new network management solutions. OpenFlow aims to provide an easy development and evaluation of new control solutions, through the centralized control and testbeds, which can be created in commercial networks. Creation of testbeds is necessary in order to develop and test new ideas on platforms that are, as much as possible, similar to the production environments. This task can be easily accomplished by FlowVisor implementation. In this paper, it has been described that multiple virtual networks can be created within a single production environment, using the FlowVisor. FlowVisor enables coexistence of multiple logical networks, which can use and make impact only to delegated resources. This provides the guaranty that virtual testbed can be created within commercial networks, with low risk that experiments could jeopardize commercial traffic.

Further OpenFlow technology penetration into commercial networks will inevitably lead to the FlowVisor implementation. That is because of the tremendous virtualization potential, not only for testbed providing within commercial networks, but also for providing new services and solving existing network's problems, that cannot be appropriately addressed by commercial technologies.

Considering the potential of network virtualization and following trends in SDN area, we have deployed OpenFlow testbed and successfully conducted its virtualization. Because of the low traffic load in created virtual networks, and compared to it relatively huge available network resources, virtualization of remaining three resources has not been demonstrated. Further work will be aimed on providing their virtualization, with the focus on bandwidth virtualization as the most challenging and interesting task. Thus, FlowVisor's ability to provide full virtualization, even in the case of traffic load that exceeds available resources, will be demonstrated.

REFERENCES

[1] Open Networking Foundation, "*Software-defined networking: The new norm for networks,*" ONF White Paper, April 2012.

[2] J. S. Turner, P. Crowley, J. DeHart, at all, "*Supercharging planetlab: a high performance, multi-application, overlay network platform.*" ACM SIGCOMM '07 New York, NY, USA, 2007.

[3] N. McKeown, et al., "*OpenFlow: enabling innovation in campus networks*", ACM SIGCOMM, Computer Communication Review, Vol. 38, Issue: 2, Mart 2008.

[4] Igor Radusinović, Miloš Odalović, "*Improving and Simplifying Control in OpenFlow Networks*", Proc. of 21th Telecommunication Forum TELFOR 2013, pp. 86-89, Belgrade, Serbia, Novembar 2013.

[5] A. Shalimov,et al.,, "Advanced Study of SDN/OpenFlow controllers", Proceedings of the CEE-SECR '13: Central & Eastern European Software Engineering Conference in Russia, ACM SIGSOFT, October 23-25, 2013, Moscow, Russian Federation

[6] OpenFlow Switch Specification v1.0. Web site: http://www.openflow.org

[7] R. Sherwood et al., "*Carving Research Slices Out of Your Production Networks with OpenFlow,*" ACM SIGCOMM Computer Communication Review, vol. 40, pp. 129–130, Januar 2010.

[8] R. Sherwood et al., "*Flowvisor: A Network Virtualization Layer,*" tech. rep., OpenFlow Switch Consortium, Oktobar 2009.

[9] R. Sherwood et al, "Can the Production Network be the Testbed" *OSDI '10*, October 2010.

[10] Balázs Sonkoly, et al., "*OpenFlow Virtualization Framework with Advanced Capabilities*," EWSDN, 2012.

[11] Hao Jin, Deng et al., "*OpenFlow-Based Flow-Level Bandwidth Provisioning for CICQ Switches.*" IEEE Trans. Computers 62(9): 1799-1812 (2013)