ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0037

Extending OpenFlow in Virtual Networks

Lorena Isabel Barona López, Ángel Leonardo Valdivieso Caraguay, Luis Javier García Villalba
Group of Analysis, Security and Systems (GASS)
Department of Software Engineering and Artificial Intelligence (DISIA)
Faculty of Information Technology and Computer Science, Office 431
Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases, 9
Ciudad Universitaria, 28040 Madrid, Spain
Email: {lorebaro, angevald}@ucm.es, javiergv@fdi.ucm.es

***Abstract*— Software Defined Networking (SDN) is a novel technology that has become a prominent topic in the last years. In any research is essential to have emulators and simulators in order to test new applications or protocols. In this context, we present the integration of OpenFlow protocol with Virtual Networks over linuX (VNX) tool, as new alternative for the emulation with SDN. VNX/OpenFlow approach integrates three kind of tools, an OpenFlow compliant switch (Open vSwitch), Network Operative Systems (POX, NOX and Beacon) and finally tools to control the performance and the network traffic. For the validation process, we present two VNX/OpenFlow scenarios to test the correctness of this tool. Finally, the result of this work allows the deployment of virtual scenarios with OpenFlow protocol.**

Keywords*— Emulation, OpenFlow, Software Defined Networking; Virtualization.

I. INTRODUCTION

Network data traffic has grown exponentially in the last years due the emergence of real time applications, video streaming, the rise of social networking, the introduction of cloud computing, among others. The research community has created protocols in order to cover these new needs, however the standardization process takes a long time and the improvements in communication methods and information processing are almost nonexistent [1].

Existing networks should have an open control and provide a real environment to tests with production traffic, due to these requirements the concept of Software Defined Networking arises [2]. SDN is not a new concept, rather is the result of many research projects such as the Active Networks and Ethane project [3]. SDN takes advantage of the best characteristics of these technologies (programmability, control and data plane separation), changing the way we see networks today. SDN allows the separation of data and control plane in network devices [4]. The control of the network behavior is in charge of an external device known as Network Operative System (NOS). The communication between network devices and the controller is established with a defined protocol, the most known OpenFlow [5].

Currently, a great number of enterprises like Google have incorporated OpenFlow in their infrastructures and devices, and there are some organizations, such as Open Networking Foundation (ONF), which promote the development and the widespread of OpenFlow and SDN architecture. There are few projects to test with SDN such as simulators, emulators or testbeds. One of the first OpenFlow testbed was developed by Global Environment for Network Innovations (GENI) [6], which interconnects the principal universities of United States. Likewise, the project OpenFlow in Europe: Linking Infrastructure (OFELIA) [7] connects 8 OpenFlow islands, allowing experimentation with this technology.

Other interesting tool is ns-3 simulator [8]. Although ns-3 has support for OpenFlow, it does not work with typical controllers such as POX [9], NOX [10], Beacon [11], Floodlight [12], OpenDaylight, and so on. Instead, ns-3 has its own OpenFlow controller. Regarding OpenFlow emulators, the most known is Mininet which is used for rapidly prototyping large networks [13]. Mininet can run real applications with a great variety of topologies; however, the performance fidelity depends on the CPU capacity and the number of the emulated hosts. Additionally, there is a hybrid approach that combines simulation and emulation in one tool called EstiNet [14]. It has not problems with fidelity performance, however, it is not a free tool.

There is a wide range of tools for experimentation with virtual networks, such as the virtualization tool called Virtual Networks over linuX (VNX) [15]. VNX is used in education and research fields, for instance in the experimentation with Intrusion Detection Systems (IDS), Multipath TCP (MTCP), among others. This paper presents the integration of this tool with OpenFlow protocol. For this purpose, OpenFlow-enabled switch and controllers are integrated.

This work has been divided into five sections, as follows: The second section contains an introduction of SDN and OpenFlow protocol. Then, the third section presents the description of simulation and emulation tools. Next, the fourth

doi:10.15849/icit.2015.0037

section shows the VNX-OpenFlow integration process and the validation of two test scenarios. Finally, a discussion is opened in the fifth section.

II. Software Defined Networking

Software Defined Networking introduces a paradigm change in the network communication, facilitating the innovation and the network programmability. SDN proposes the separation between the control and the data plane in networking devices. Consequently, the network is more flexible, programmable and it has automation capabilities. The own device could carry out advanced capacities such as firewall rules, load balancing, among others.

The control of whole network is performed by a central point known as a controller. The network devices are connected with the controller through secure communication channel like Sockets Secure Layer (SSL). In the communication process is needed a standardized protocol the most known OpenFlow [5], which defines the communication rules between controller and OpenFlow compliant switches. OpenFlow offers new features that enable experimentation without expose the internal structure of switches from different vendors. For this purpose, OpenFlow delimits the basic functions of OpenFlow switches based on common characteristics of traditional Ethernet switch. OpenFlow defines three kind of tables, these are: flow, group and meter table. OpenFlow also introduces the flow concept, which can be defined as a kind of traffic such as the http requests, traffic to the same destination address, and so on. Moreover, OpenFlow establishes a pipeline in order to process the incoming packets. The packet is first matched against flow entries of flow table 0 and may continue with the next tables, depends on the result of the match in the table. Flow entries match packets based on the priority field (highest priority). If a flow entry is found, the instructions are executed (Modify packet and update match fields, update action set, update metadata). If the packet does not match with a flow entry in any table, the outcome depends on the configuration of the table miss. A possible action is to search in the next table. Based on the SDN architecture and the business requirements many tools have been developed, such as:

- Virtualization tools [16].
- Network Operating System (controllers) [9] [10].
- Virtual switches [17].
- Tools for Quality of Services and Quality of Experience [18].
- Management [19] [20].
- Optical Networks [21] [22].
- Traffic engineering and load balancing [23].
- Load Balancing [24].
- Simulation and Emulation tools [8] [13] [14].

All of these research fields are deployed and tested through some approaches; real testbeds, emulator or simulators [25]. OpenFlow testbeds [6] [7] allow the experimentation in real environments on a large scale. However, testbeds are not easily accessible by potential researchers. For its part, simulation and emulation approaches provide facilities in terms of scalability, portability and accessibility in the case of open source tools. Nevertheless, in some cases they produce inaccurate outcomes. We describe some familiar tools ns-3, Mininet and EstiNet, as well as VNX/OpenFlow.

III. Simulation and Emulation Tools

NS-3 is a simulator tool focuses on research and educational fields. It is an open sources simulator that provides an extensible network platform with several external animators, data analysis and visualization tools. In order to enable the simulation with OpenFlow protocol, Ns-3 implements its OpenFlow-enabled switch and its own controller, as a modules written in C++. The switch component is known as *OpenFlowSwitchNetDevice*. This object consists of a set of net devices that represent the switch ports, according to the OpenFlow Switch Specification v0.8.9. Even though Ns-3 can be used for real-time simulations, there are some issues that the user should take into account such as the slow learning curve to use the tool, the compatibility with a basic OpenFlow version (0.89) and specially it does not run a typical OpenFlow controller. Therefore, the controller applications generated with ns-3 controller cannot be used in real network. If a controller like Pox or Floodlight was required, these will need substantial modifications.

For its part, Lantz et al. in [13] proposes Mininet, a virtualization tool for rapidly prototyping large networks in a single laptop. This tool includes OpenFlow support and combines lightweight virtualization capabilities over Linux operative system with an extensible CLI and API. A scenario built with Mininet is deployable, interactive, scalable, realistic and it can easily share. In fact, the Mininet topologies and the controller applications can be used for others researchers without modifications in the emulation environment as well as in real networks. Mininet run on virtual machine monitors like VMWare, XEN and VirtualBox or it can be installed in a Linux system. Mininet allows the deployment of hundreds of nodes, emulating OpenFlow-enabled switches, controllers like POX, virtual links and hosts. Mininet shares components like the file system, the user ID space, the kernel, device drivers, among others. Tough, Mininet is the most popular tool for SDN has limitations of performance fidelity related with the available resources, real bandwidth and the timing of the process.

A novel hybrid approach has recently presented called EstiNet [14]. This combines the best characteristics of both simulation and emulation mode in one tool. On the one hand, it allows the deployment of large networks in a flexible, easy, scalable and repeatable way. On the other hand, EstiNet takes into account the timing needs for real applications in order to obtain the same results in both, virtual and real deployments. EstiNet supports 1.3.2 OpenFlow Switch Specification and it can run NOX, POX, Floodlight, and Ryu controllers without any modifications. For this purpose, EstiNet intercepts the packets between two real applications through tunnel network interfaces and redirects the packets to the EstiNet simulation engine. The entire process is based on a simulation clock, which allows accurate results. Besides, EstiNet provides a graphical user interface for configure the scenarios and observe

doi:10.15849/icit.2015.0037

the outcomes from the simulations. The results of this tool show better scalability and performance than Mininet, however their main problem is that it need a payment for the tool. The universities can embrace the EstiNet University Program. This grants a license during six months with a cost of US$1500 or a license to 12 months for US$2500, becoming its main disadvantage.

As we have seen, there are few tools or testbeds that allow the SDN experimentation. We present VNX a modular architecture based on plugins (Fig. 1), which allows the deployment of virtual testbeds. This tool includes the code of the previous tool VNUML [26].

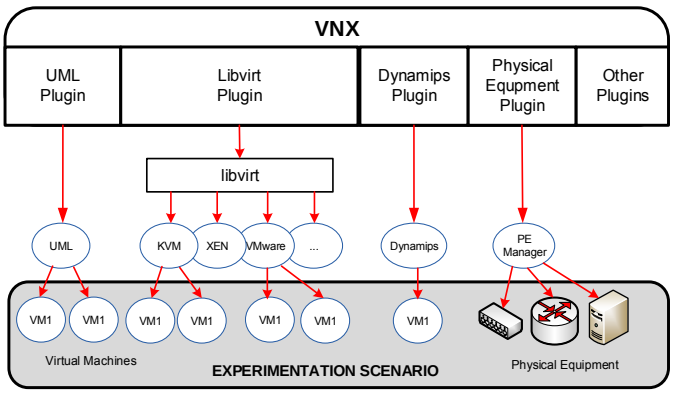

Fig. 1. VNX Architecture [15].

The plugins used by VNX are:

- UML (User Mode Linux) can be considered a hypervisor-based technique.
- libvirt allows virtualization capabilities and some virtualization platforms, such as Xen, VMware, KVM, VirtualBox, etc.
- Dynamips plugin allows the emulation the hardware of Cisco routers.
- Olive allows the integration of Juniper routers.
- Physical equipment plugin, which allows the connection between VNX physical islands.

VNX is a free tool based on Linux that allows the easy creation and management of large virtual scenarios over a single server or a cluster. The scenarios can have nodes in some physical hosts and can use different operative systems, for example Linux and Windows. In turn, each physical host can deploy their own virtual testbed. Besides, VNX allows the creation of large scenarios with hundred or even thousands of virtual machines. This process uses the copy on write technique (cow), which starting the virtual machines from a single image file known as filesystem. In this way, the nodes can share the same filesystem. The filesystem is mounted in read-only mode. If a virtual node is modified, the differences are stored in a private filesystem.

VNX is also focused on education and research. In [15] a large virtual network scenario was created. It is a laboratory for dynamic routing that involve 44 virtual devices (16 Cisco routers, 6 Juniper routers, 6 Linux/Quagga routers, 12 end user and 4 Servers). This testbed is a typical scenario deployed with VNX and shows its potential.

One of the main SDN challenges is the integration of heterogeneous networks. VNX could provide the ideal environment to combine OpenFlow-enabled islands and legacy networks. The integration process is described in the next section.

IV. Integration and Validation

VNX should be implemented over a Linux operating system. The guidelines for configuration, modifications and filesystems are available in the official site of VNX project [27]. In order to testing with OpenFlow protocol, VNX needs the integration of some critical elements, an OpenFlow-enable switch for virtualization environments and a network operative system for network control. Additionally, it would be useful the integration of performance tools or data traffic analyzer. VNX was installed on a physical host with Ubuntu 12.04. Then, we choose two different filesystems. For controller device is desirable a graphical interface (ubuntu-12.04-gui-v024) to analyze the traffic. The second filesystem is a console interface (ubuntu-12.04-v024), which is used for simulated hosts and routers. The graphical filesystem was modified to make the controller functions, 3 of them were integrated: POX (based on Python) which is one of the most widely used today, NOX based on c++ and Python and finally Beacon which uses Java. The integration and configuration process are available in the official sites of each project. Additionally, in order to improve the functionalities of VNX/OpenFlow, three tools were installed: Wireshark, tcpdump and iperf. The wireshark tool is indispensable because originally it does not identify OpenFlow traffic. For this purpose, a dissector plugin for OpenFlow must be compiled and installed in the filesystem. Dissector allows to decode all information of specific incoming packets, in this case OpenFlow (version 1.0). Other important changes is the integration of Open vSwitch (OVS) [17]. OVS is an open source tool that allows the creation of switches in virtualization environments. OVS matches the virtual machines, providing better performance than the traditional bridge, such as VLANs, netFlow, QoS, bonding, mirroring, among others. OVS works transparently with VNX, for both legacy and OpenFlow networks. The version used in this paper is 1.4.0. After we create the .xml specification (Fig. 2).

```
<vm type="libvirt" name="C2" os="linux" subtype="kvm">
    <filesystem type="cow">/usr/share/vnx/filesystems/rootfs_ubuntu-gui</filesystem>
    <mem>512M</mem>
    <console display="yes" id="0"/>
    <console display="no" id="1"/>
  - <if id="2" net="Net1">
        <ipv4>10.0.1.3/24</ipv4>
    </if>
    <route type="ipv4" gw="10.0.1.1">default</route>
</vm>
<host>
  - <hostif net="Net2">
        <ipv4>10.0.2.2/24</ipv4>
    </hostif>
    <route type="ipv4" gw="10.0.2.1">10.0.0.0/16</route>
</host>
```

Fig. 2. XML Specification for Design Phase.

doi:10.15849/icit.2015.0037

Once we have the file with .xml specification, the virtual scenario is deployed and matched with the controller. For the validation process we replicate the topology of OpenFlow Tutorial, as a point of reference to see the VNX operation. This tutorial was developed by Stanford University [28] and it deploys a topology (subnet 10.0.0.0/24) with 3 virtual hosts (h2, h3 and h4), an OpenFlow switch (s1) and one controller (c0). Two scenarios are presented: a basic scenario (Fig. 3a) that is identical to OpenFlow Tutorial and the second scenario incorporates more subnets and a second controller (Fig. 3b).

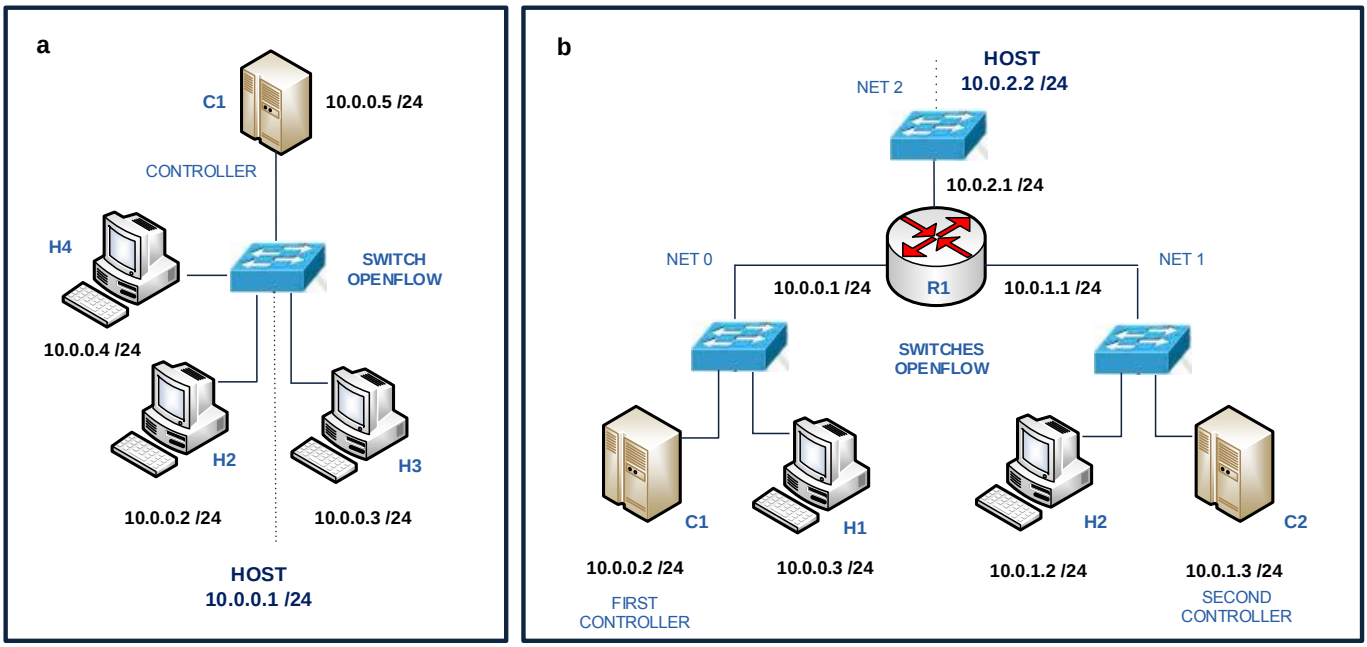

Fig. 3. (a) Scenario 1. Basic Scenario; (b) Scenario 2. Scenario with two Controllers.

The first scenario (Fig. 3a) has an OpenFlow-enabled switch and four hosts (C1, H2, H3, H4), all of them with Ubuntu 12.04. H2, H3 and H4 work with textual consoles and the controller (C1) works with a graphical console. The second scenario (Fig. 3b) is formed by five Ubuntu 12.04 virtual machines (router and hosts work with textual consoles and controllers with graphical console) according to the following structure:

- 3 switches in different subnets (Net0: 10.0.0.0/24, Net1: 10.0.1.0/24 y Net2: 10.0.2.0/24).
- 2 controllers (C1: 10.0.0.2 and C2: 10.0.1.3).
- 2 hosts (H1:10.0.0.3 and H2:10.0.1.2) each one in different subnets.
- Subnets communicate through the router (R1).

The proofs of concept of this work were made exclusively with Ubuntu virtual machines, but it is possible to use another kind of operating system. Data traffic was analyzed with Wireshark. At first, Wireshark shows only typical protocols, such as ICMP, UDP, IP, among others, because OVS works as an Ethernet switch by default.

In order to enable OpenFlow traffic, OVS must be connected with the controller. There are two configuration modes, which determine the switch behavior for a controller fail condition. These modes are:

- Fail standalone: The default configuration mode. If OVS does not receive the inactivity probe interval three times, the OVS takes the control of the switch and it works like a normal Ethernet switch (MAC-learning switch). When the connection is lost, the switch handles the incoming packets using the OFPP_NORMAL reserved port. Moreover, the switch will attempt to connect with the controller. These mode is usually available in OpenFlow hybrid switches.
- Fail secure: In this mode the OVS cannot take the network control if the controller fails. The network will be uncommunicated during the failure. Then, OVS will attempt to connect with the controller, until obtain a response. This mode is commonly used to avoid forwarding loops.

Once the communication is established, the controller (or controllers) should maintain the links with all switches. There are three kinds of roles for the connection. The default role is OFPCR_ROLE_EQUAL and it allows full control over the network. The second role is known as OFPCR_ROLE_SLAVE, in which switches are configured in read only mode, therefore the controller has limited control. The third role, OFPCR_ROLE_MASTER works in the same way that OFPCR_ROLE_EQUAL, but there is only one controller with this role, other controllers are changed to slave role. In the second scenario all switches are connected with C1 and C2 controllers in EQUAL role. In this way, we provide redundancy to the second scenario.

Proofs were made with standalone and secure mode in both scenarios. We used POX controller with three applications,

doi:10.15849/icit.2015.0037

forwarding.l2_learning, forwarding.l3_learning and forwarding.hub. Additionally, we wrote scripts in order to automate the process. These scripts contain the code for the deployment of the above mentioned scenarios and the establishment of links between switches and controller.

In both scenarios data traffic was generated with ICMP and web requests between the hosts of the topologies. OFP (message for the establishment of network communication), OFP-ARP, OFP-ICMP (packet-in, packet-out) messages were captured with Wireshark analyzer and tcpdump tools as shown in Fig. 4.

26	4.088734	10.0.0.2	10.0.2.2	OFP	74	Echo Reply
27	4.088827	10.0.0.2	10.0.2.2	OFP	74	Echo Reply
28	4.101774	02:fd:00:00:03:01	02:fd:00:00:01:01	OFP+ARP	126	Packet In
29	4.105569	10.0.0.2	10.0.2.2	OFP	90	Packet Out
32	4.106380	02:fd:00:00:01:01	02:fd:00:00:03:01	OFP+ARP	126	Packet In
33	4.108450	10.0.0.2	10.0.2.2	OFP	90	Packet Out
37	4.425297	10.0.0.3	10.0.1.2	OFP+ICMP	182	Packet In
39	5.038602	10.0.1.2	10.0.0.3	OFP+ICMP	182	Packet In
41	5.425545	10.0.0.3	10.0.1.2	OFP+ICMP	182	Packet In

Fig. 4. Traffic Capture Scenario 2.

Fig. 4 shows an ICMP proof from host h1 (10.0.0.3) to host h2 (10.0.1.2) performed in the second scenario, with the component forwarding.l2_learning of POX controller and in standalone mode.

Both scenarios work properly with OpenFlow protocol, however in second scenario there were duplicated messages (from controllers C1 and C2). This is because OpenFlow does not define coordination mechanisms among controllers in the same network or in different domains [29]. At present, this process is done with other components. For instance, Fonseca et al. in [30] introduces the CPRecovery component, which allows keeping the consistency between the primary and backup controllers. This component provides seamless transition between the primary and secondary controller through two steps, the replication phase (maintain updated data) and the recovery phase. The replication phase acts during the normal network behavior and the recovery phase acts in case of failure. Another challenge in large topologies is the communication among controllers in different SDN domains. At the present time, Internet Engineering Task Force (IETF) is working in a standard called interfacing SDN Domain Controllers (SDNi) for exchange routing information (network topology views, network conditions, event reports) and application requirement.

A general overview for the whole process in order to interact with VNX/OpenFlow scenarios is shown in figure 5. The first phase consist of the design and creation of VNX scenarios based on .xml specification. The second phase is related to the deployment or destruction of these scenarios through specific commands (vnx –f –v --create). Then, the controller must be connected with the switches and the user should configure the operation mode (standalone, secure, equal, slave, master). The user can create their own topologies and programs with the controller and finally can interact with the OpenFlow testbed.

V. Conclusion and Discussion

This work presents the integration process between VNX tool and OpenFlow protocol. The filesystems used by virtual machines and nodes was modified. We create a SDN environment through the integration of two main components: an OpenFlow compliant switch (Open vSwitch) and three network operating systems (NOX, POX and Beacon). Besides the controller has incorporated some performance and analyzer tools, these are Wireshark, tcpdump and iperf. Proofs of concept were carried out with POX components and two configuration modes (secure and standalone).

We can verified the exchange of OpenFlow messages (OFP+ARP, OFP+ICMP Packet In, OFP packet Out) with Wireshark analyzer. Although in the validation process we only used Ubuntu, future proofs can use multiple operating systems such as Windows. Now the user can create their own topologies and controller programs in order to experiment with OpenFlow protocol and SDN technology, which was the main objective of this work.

Today, VNX allows the deployment of large and complex OpenFlow networks in distributed environments. VNX allows not only the deployment of virtual scenarios in a single laptop, but also allows the inclusion of physical equipment (each one can have its own scenario with virtual machines), that is, VNX works in distributed scenarios. This is the main contribution of VNX over Mininet, since the communication between two scenarios in Mininet is a complex process. In this way, VNX enable the communication between OpenFlow networks and legacy networks that is one of the main challenges of SDN, the transition and migration process between heterogeneous networks. Besides, take into account that virtual scenarios may include Cisco and Juniper devices, therefore inside the virtual scenarios we could test with OpenFlow and no OpenFlow networks. Moreover, VNX allows the easy experimentation with specific services such as multimedia applications, deployment of servers, among others. The developer can customize the filesystem of the hosts and in this way, testing their new ideas and applications.

VNX also allows another kinds of operating systems for the virtual machines, such as Debian, Windows and Fedora. This is another strong point compared with Mininet, which uses only a Linux kernel. If a user want to test a Windows application over an OpenFlow network, the windows filesystem may include the application code.

doi:10.15849/icit.2015.0037

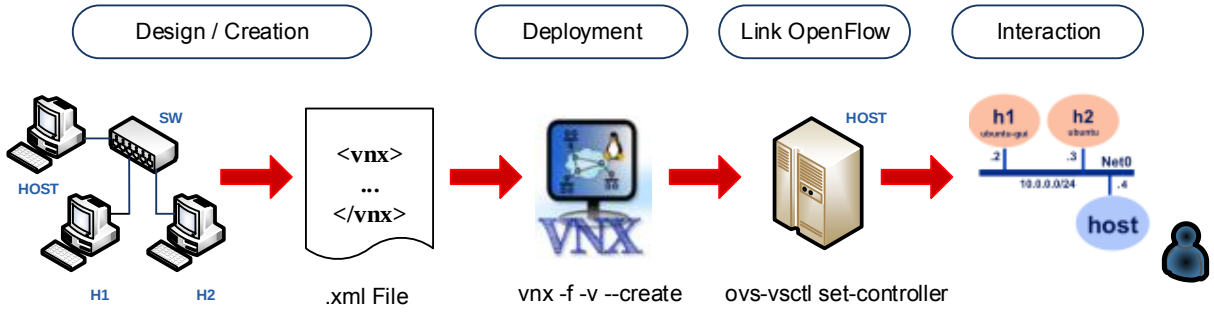

Fig. 5. Workflow of VNX/OpenFlow.

ACKNOWLEDGMENT

The research leading to these results has been partially funded by the European Union's H2020 Program under the project SELFNET (671672). Lorena Isabel Barona López and Ángel Leonardo Valdivieso Caraguay are supported by the Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación SENESCYT (Quito, Ecuador) under Convocatoria Abierta 2012 and 2013 Scholarship Program. This work was partially supported by the "Programa de Financiación de Grupos de Investigación UCM validados de la Universidad Complutense de Madrid – Banco Santander".

The authors would like to thank to David Fernández Cambronero for his comments and suggestions about VNX tool and Ana Lucila Sandoval Orozco for her valuable comments and suggestions to improve the quality of the paper.

REFERENCES

[1] W. Stallings, "Software Defined Networks and OpenFlow," *in The Internet Protocol Journal*, vol. 16, no. 1, March 2013, pp. 2-14.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *in ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, April 2008, pp. 69-74.

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, "Ethane: Taking Control of the Enterprise," *in Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, USA, August 2007, pp. 1-12.

[4] A. L. Valdivieso Caraguay, L. I. Barona López, L. J. García Villalba, "Evolution and Challenges of Software Defined Networking," *in Proceedings of the Workshop on Software Defined Networks for Future Networks and Services*, Trento, Italy, November 2013, pp. 47-55.

[5] O. S. Consortium et al., "OpenFlow Switch Specification v.1.3.4," March 2014 pp. 1-171.

[6] C. Elliott, "GENI: Opening Up New Classes of Experiments in Global Networking," *in IEEE Internet Computing*, vol. 1, January 2010, pp. 39-42.

[7] M. Suñé, L. Bergesio, H. Woesner, T. Rothe, A. Köpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda, M. Kind, T. Dietz, A. Autenrieth, V. Kotronis, E. Salvadori, S. Salsano, M. Körner, S. Sharma, "Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed," *in Computer Networks*, vol. 61, March 2014, pp. 132-150.

[8] T. R. Henderson, M. Lacage, G. F. Riley, "Network Simulations with the ns-3 Simulator," *in Proceedings of the ACM SIGCOMM'08*, Seattle, WA, USA, August 2008, pp.17-22.

[9] POX, https://openflow.stanford.edu/display/ONL/POX+Wiki.

[10] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, "NOX: Towards an Operating System for Networks," *in ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, July 2008, pp. 105-110.

[11] D. Erickson, "The Beacon Openflow Controller," *in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, New York, NY, USA, August 2013, pp. 13-18.

[12] Floodlight project, http://www.projectfloodlight.org/.

[13] B. Lantz, B. Heller, N. McKeown, "A Network in a Laptop: Rapid Prototyping for. Software-Defined Networks," *in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, New York, NY, USA, October 2010, pp. 1-6.

[14] S. Y. Wang, C. L. Chou, C. M Yang, "EstiNet OpenFlow Network Simulator and Emulator," *in IEEE Communications Magazine*, vol. 51, no. 9, September 2013, pp. 110-117.

[15] D. Fernández, A. Cordero, J. Somavilla, J. Rodriguez, A. Corchero, L. Tarrafeta, F. Galán, "Distributed Virtual Scenarios over multi- Host Linux Environments," *in Proceedings of the 5th IEEE International DMTF Academic Alliance Workshop on Systems and Virtualization Management*, Paris, France, October 2011, pp. 1-8.

[16] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, "Flowvisor: A Network Virtualization Layer," *in Technical Report OpenFlow Switch Consortium*, October 2009, pp. 1-15.

[17] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, S. Shenker, "Extending Networking into the Virtualization Layer," *in Proceedings of the Eight ACM Workshop on Hot Topics in Networks*, HotNets-VIII, HOTNETS '09, New York City, NY, USA, October 2009, pp. 1-6.

[18] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, P. Dely, "Towards QoE-driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking," *in Proceedings of the 20th IEEE International Conference on Software, Telecommunications and Computer Networks*, (SoftCOM), Split, Croatia, vol. 1, September 2012, pp. 1-5.

[19] R. Bennesby, P. Fonseca, E. Mota, A. Passito, "An Inter-AS Routing Component for Software-Defined Networks," *in Proceedings of the IEEE Network Operations and Management Symposium*, Maui, Hawaii, USA, April 2012, pp. 138-145.

[20] F. Farias, J. Salvatti, E. Cerqueira, A. Abelem, "A Proposal Management of the Legacy Network Environment Using Openflow Control Plane," *in Proceedings of the IEEE Network Operations and Management Symposium*, Maui, USA, April 2012, pp. 1143-1150.

[21] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, L. Ong, "Packet and Circuit Network Convergence with OpenFlow," *in Proceedings of the IEEE Conference on Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference (OFC/NFOEC)*, San Diego, CA, USA, March 2010, pp. 1-3.

[22] M. Channegowda, R. Nejabati, M. Rashidi Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martínez, "First Demonstration of an OpenFlow based Software-Defined Optical

doi:10.15849/icit.2015.0037

Network Employing Packet, Fixed and Flexible DWDM Grid Technologies on an International Multi-Domain Testbed," *in Proceedings of the European Conference and Exhibition on Optical Communication*, Amsterdam, Netherlands, September 2012, pp. 1-3.

[23] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, R. Johari, "Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow," *in Proceedings of ACM SIGCOMM Demo*, Barcelona, Spain, August 2009, pp. 1-2.

[24] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou. "A Roadmap for Traffic Engineering in SDN-OpenFlow Networks". *in Computer Networks*, vol.71, June 2014, pp. 1-30.

[25] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Reijendam, P. Weissmann, N. McKeown, "Maturing of OpenFlow and Software-defined Networking through Deployments," *in Computer Networks*, vol. 61, March 2014, pp. 151-175.

[26] F. Galán, D. Fernández, W. Fuertes, M. Gómez, J. E. L. de Vergara, "Scenario-based Virtual Network Infrastructure Management in Research and Educational Testbeds with VNUML," *in Annals of Telecommunications-Annales des Telecommunications*, vol. 64, May 2009, pp. 305- 323.

[27] "Virtual Networks over linuX (VNX)," http://web.dit.upm.es/vnxwiki/index.php/Main Page.

[28] OpenFlow Tutorial, http://www.openflow.org/wk/index.php/OpenFlow Tutorial.

[29] A. L. Valdivieso Caraguay, A. Benito Peral, L. I. Barona López, L. J. García Villalba, "SDN: Evolution and Opportunities in the Development IoT Applications," *in International Journal of Distributed Sensor Networks*, vol. 2014, May 2014 pp. 1-10.

[30] P. Fonseca, R. Bennesby, E. Mota, A. Passito, "A Replication Component for Resilient OpenFlow-based Networking," *in Proceedings of the IEEE Network Operations and Management Symposium, Maui*, Hawaii, USA, April 2012, pp. 933-939.