Open Source Tool for Networks Management Communication

Nuno Tiago Louro Simões
School of Technology and Management
Polytechnic Institute of Leiria
Leiria, Portugal
E-mail: 2130967@my.ipleiria.pt

Carlos Manuel da Silva Rabadão
Research Center for Informatics and Communications
Polytechnic Institute of Leiria
Leiria, Portugal
E-mail: carlos.rabadao@ipleiria.pt

***Abstract*—Considering the complexity of the networks, one of the solutions for this complexity could be to centralize its configuration. Thus the *Software-Defined Networking* (SDN) concept may be an important solution. This paper suggests the implementation of a tool to support the development and testing of networks and services before they are put into production. The use of a tool that simplifies the configuration of a network service makes the networks and services to be less susceptible to errors and failures by those who set them up, thus allowing telecom operators, among others, to be able to create new services, improve the monitorization of their human resources and, above all, improve their financial results. In the end, success will be achieved because with a simple interaction and basic knowledge we are able to manage network services.**

Keywords - SDN; network services; network programming; NSO.

I. INTRODUCTION

The number of electronic devices with Internet access has been increasing in recent years [1]. Nowadays, it is even possible to have Internet access with a simple watch. With the appearance of these devices along with the advances in Information technology (IT), telecom operators need to introduce new features to capture the customer's attention. One of these innovations could be the creation of new services in the network. One of the problems that the creation of new services currently faces is the congestion that the network has. This makes the configuration of networks complex and increases the difficulty in creating new services. Nevertheless, operators have been able to manage both the network and the services, but it is natural that they are susceptible to failure by those who manage and implement them. This process is typically done by a human. Most failures stem from several factors, including pressure caused by the need to put new services quickly on the market or by the routine repetition of processes that limit the potential of the network [2].

We can hardly develop a perfect *software* immune to failures and errors, but there are methods that can be used to try to prevent them, for example, the use of *scripts*. *Scripts* allow us to automate some tasks. As these *scripts* are developed by humans, they will be susceptible to failures and errors, even if they are unintentional.

Taking into account the foregoing considerations, the scientific community has been looking for new approaches that can help to reduce limitations. This area is explained in the next sections. Considering the increasing number of people using devices with internet access and the consequent increase of the network complexity, we are motivated to develop an application to help in the service and network management so that it can be innovated and improved. The aim of this paper is to present the development of a tool, based on the concept of SDN, which allows the testing of a network and the implementation of services before they are produced.

Concerning the management of services, one of the approaches associated to it is the SDN concept. SDN is the basis of this work, which will be introduced in Section II. In this section we will present some SDN solutions existing in the market, as well as some of the technologies used. In Section III, we will present the proposed architecture that supports this work. The architecture contributes to the mitigation of previously presented problems. In Section IV, we will explain the implementation of our proposal. Finally, in the last section, we will present the conclusions and suggestions to work.

II. SOFTWARE-DEFINED NETWORKING

This section initially presents some concepts for a better understanding of the article. After, we will introduce some commercial SDN solutions that exist in the market and some technologies used for the implementation of the tool created.

A. Background

According to the *Open Networking Foundation*, the SDN is the physical separation of the control plane and the forwarding plane of the network [3]. With SDN concept, the networks will be configured and managed in a centralized way [4], facilitating the development of new *standards* and services. The SDN concept emerged at the same time as other technological solutions, from which the need motivated by complexity in the network arises. These needs combined with the fact that operators need to put more services in the market, as soon as possible, turn the process more complex and more likely to fail.

The purpose of SDN is to make the management of the network easier and transform the network programmable [5]. Thus, it simplifies the understanding of the network, which means that operators can do their job quicker and easier, according to the *time-to-market's* factor. Consequently, the operators may have good financial profits, which is an advantage.

Now we will present some of the existing SDN solutions: *Virtualized Services Controller* (VSC), by the internal company of Alcatel-Lucent, the Nuage Networks [6] [7] and

 ISBN: 978-1-61208-482-4

Network Control System (NCS) [8] by Tail-f, currently owned by Cisco portfolio.

B. SDN solutions

In this subsection we will make a brief analysis of each SDN solutions studied.

1) Virtualized Services Controller (VSC)

VSC, based on *Alcatel-Lucent Service Router OS* [9], is the SDN solution control panel of Nuage Networks and the most powerful SDN controller in the industry [7] [10]. VSC works in similar way to the network control plane for the data center, because it has a complete view of the network and its services. VSC automatically discovers network parameters, whatever type they are: Layer 2 (*switching*), Layer 3 (*routing*), *Quality of Service* (QoS) or security rules. In the VSC, the connection between the controller and the network routing is established through the communication protocol - *OpenFlow* [11]. This protocol allows the communication between the service controller and the network layer where it should find the hardware, i.e., the *hypervisor* and *vSwitch* [12].

2) Network Control System (NCS)

The NCS is the solution to control the network established by Tail-f. Later Cisco acquired Tail-f Company and the name of the SDN solution set was changed to "*Cisco Network Service Orchestrator* (NSO) *enabled by Tail-f*" [8]. The NSO is nothing more than a transparent layer, or interface, for those who configure the network. The NSO was meant to facilitate the creation and configuration of network services [13]. This solution is independent of brands and network equipment manufacturers, whether it is real or virtual. This SDN solution can be used to interact with both users/network administrators as well as with management applications that are already used in a network.

To sum up, all SDN solutions up to now are more or less similar. They are all are composed by three parts: implementation, monitoring and infrastructure/network equipment. This structure is more or less predictable given the SDN architecture.

C. Technologies used

This subsection will refer briefly to some technologies used or associated with the development of the proposed solution and also related to SDN. These technologies are: YANG, *extensible Markup Language* (XML) and *Network Configuration Protocol* (NETCONF).

1) YANG

The YANG is a data modelling language used for a data state configuration model. This language is used by the network configuration protocol - NETCONF - and is published in the *Request for Comments* (RFC) 6020 of September 2010. The YANG is related to the content and operations in layers of NETCONF [14].

2) XML

The XML is used to describe data. This shape can be easily used to read and write data. XML is adopted in many areas of information technology, including networks. It can be dynamic and it is very similar to the *Hypertext Markup Language* (HTML). We can consider that the construction of XML is done by blocks which are identified by *tags* [15].

3) NETCONF

The NETCONF is generically used to make the management of network devices configuration and it is based on the encoding in XML [16]. This protocol defines basic operations that are equivalent to commands to be executed from the *Command-Line Interface* (CLI). As in XML, NETCONF also uses *tags*. One of the manufacturers that uses NETCONF on its devices is Juniper Networks [17].

III. ARCHITECTURE PROPOSAL

In order to frame the solution/tool to propose, first we must present a logical structure of the SDN and after we will present the generic architecture of the solution developed.

The logical structure of the SDN, based on the same technology architecture, has three main layers, displayed in Figure 1 that are: *Application Plane*, *Control Plane* and *Data Plane*.

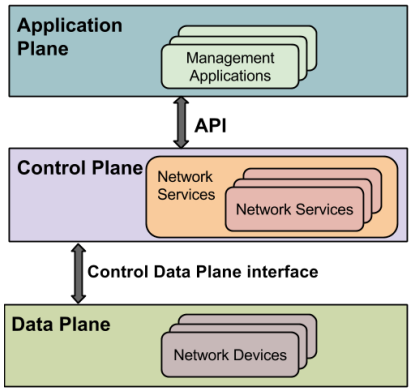

Figure 1. Logical structure of SDN

Next we will explain each layer mentioned above [18]:

- *Application Plane*: it can refer to some *net apps* such as orchestration applications, business applications and SDN applications;
- *Control Plane*: it aims to implement all coordination protocols that are necessary for the proper functioning of the *Data Plane*;
- *Data Plane*: it serves to analyze the headers of incoming packets and forward these packets to their final destination, depending on the routing and switching tables.

After presenting generically the SDN architecture, it is time to present an approach to SDN, more dedicated to network management, adopted to implement this work. The architecture shown in Figure 2 is quite simple, as it is divided into three layers: user, orchestration of the network and, finally, the network itself.

 ISBN: 978-1-61208-482-4

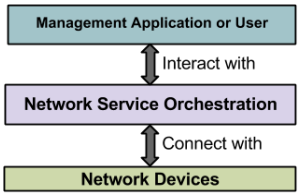

Figure 2. Generic architecture implementation performed

The architecture consists of three layers, described below:

- *Management Application or User:* this layer, as the name implies, is where the user, who will interact with the network, has the primary role and where we think he will spend most of the time;
- *Network Service Orchestration*: this is the "smart" layer of the presented architecture. In this layer the entire process will be unfolded. The Network Service Orchestration will interpret the user's *input* and transform it so that it can be applied to the network, which is the next and last layer to be presented;
- *Network devices*: this last layer is the physical infrastructure of the network. It is composed by the *core* and the access network, where it intends to apply the settings for network management and for the creation of services.

After presenting the generic architecture of the solution implemented, we will make a deeper analysis of the same.

A. Architecture used in the implementation

A more detailed architecture proposed for the development of this work is shown in Figure 3.

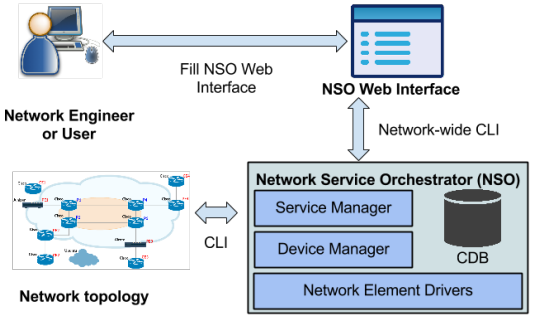

Figure 3. Architecture used in the proposal

In this figure we can observe that from the starting point (*Network Engineer or User*) to the end point (*Network Topology*), the user only interacts with a *WebUI* to configure the network mode as required. The *WebUI* is the point we have recreated, being more intuitive, specific and simpler to use, which is something new, compared to the existing tool. The novelty consists in the communication between the NSO and a *web interface*, as it is made through the Network-wide CLI and, as we can observe in the figure, this communication cam be bidirectional. The necessary mechanisms to convert the high-level user-made settings must be previously configured and implemented, allowing users with low technical level to proceed with the configuration of the network and services. Then the form communicates with the Network Service Orchestrator through the implementation made in *back-end* of *WebUI* and in command line. Note that this process is abstract to the final user. It is in the stage of communication between the NSO and the type of Network that all the fundamental processes for the correct operation of this took are taken. The NSO is divided into four parts (three layers and a part relating to data storage) [13]:

- *Service Manager*: this is where the intelligence of the NSO tool is. This layer enables the operator to manage high-level aspects of the network that are not supported by the devices that are directly connected to it. The services should be defined previously. It is from here that the management (creation, editing or deletion) of network services will be made;
- *Device Manager*: its function is to manage the configuration of transactional devices, supporting the synchronization feature of bi-directionally settings and refined changes in real time;
- *Configuration Database* (CDB): it is here that the information on the device configurations is all stored, so there is data synchronization. It is in the CDB that the synchronization, consistency and reconciliation with respect to the configuration between the services and devices occurs;
- *Network Element Drivers* (NED): they are responsible for the link between the NSO and network devices. The NED uses the concept of atomicity, i.e., the execution of a command is either correct and runs, or if a simple thing is wrong, nothing will be executed. The NSO, according to the device we want to configure, informs the device type (*device-type*) of what to do, independently of the brand/device manufacturer. The device interface is modeled on files, using the YANG, and each file is modeled with the controls - that can be updated – in the respective device. The philosophy of the NED varies from device to device. For Cisco and Alcatel, commands are converted to CLI to run on the device terminal. For the Juniper equipment, that already uses NETCONF - based encoding in XML -, the philosophy is different, i.e., not needing to convert settings.

As it was said before, the communication between NSO and the devices should be done by OpenFlow, NETCONF, XML, CLI or any other. If we do a deeper analysis of the communication, we will notice that the communication between the NSO and the network equipments are the responsibility of the NED or the OpenFlow controllers, as we can see in [19] document. Note that this communication is made by the NSO and it was not changed in the proposed tool.

We finally get to the network and the devices, which may be of different brands and models. In this solution, the NSO gets to know the equipment by means of the communication Secure Shell (SSH) protocol.

After the presentation of the proposed tool architecture, we will explain, in the next section, how it is implemented.

 ISBN: 978-1-61208-482-4

IV. PROTOTYPE

The implementation of this tool is based on the architecture presented in Section III. In this section, we will deepen the architecture used, namely the implementation carried out and which ultimately resulted in the presentation of a simple tool that makes the network services management.

A. Prototype implementation

As mentioned above, our aim is to develop an *Open Source* tool where we can test the settings of a network and its services. The network can be either real or virtual. The concept behind the tool is SDN. With this kind of tool, the entire configuration process is centralized and this same configuration does not require in-depth knowledge of computer networks. So we can simplify the configuration and understand a network. From a purely visual point of view, the developed tool is nothing more than a *Graphical User Interface* (GUI) or *WebUI*. Next, we will explain the process of implementing this tool. The solution developed is based upon three main stages:

- Scenario/network topology – where the network equipment is included;
- Development of the intermediate layer – a layer that will make the connection between the configuration and network equipment and which is transparent to the user. The development basis was the use of the platform "*Cisco Network Service Orchestrator enabled by Tail-f*" and this is the platform that connects the network topology to the graphical interface. Cisco NSO is an orchestration technology that is based on the SDN concept, since the Orchestrator Apps are part of the Application Plane, one of the layers that belong to SDN. This phase will be the *back-end* for the user;
- Graphical User Interface - primary site of interaction between the user and the network. *Front-end* for the user.

The implementation of these three stages will be presented in the following subsections.

1) Scenario/network topology

Initially a virtual Linux Ubuntu machine was created to run the 14.4 version. In this machine a network was developed on a network simulation software GNS3 [20], shown in Figure 4, where several different manufacturers were set, including Cisco and Juniper.

In Cisco's *routers* they used the file "c3725-adventerprisek9-mz.124-25d.bin" to virtualize the IOS. This model was the only one to which we had access, although we know that there are more recent models. As for Juniper, we had to use a vSRX *Open Virtual Application* (OVA) image, more specifically a 12.1X47-D15.4 version of JunOS vSRX. The only settings made in this equipment were addressing, routing, the *Open Shortest Path First* (OSPF) in this case, and the communication protocol configuration used – SSH.

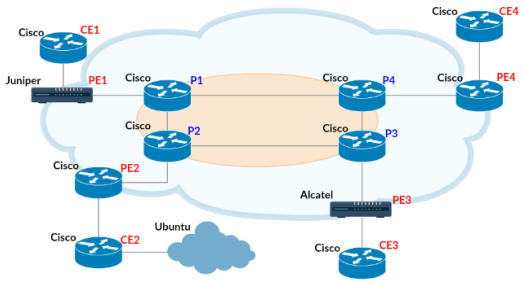

Figure 4. Network topology defined to test developed in GNS3

To bridge the gap between the topology developed and the GUI we used, as mentioned above, the NSO solution that we will explain in detailed in the next subsection.

2) Development of the intermediate layer

After the topology and configuration of the devices is completed, we have defined some services to be implemented and tested on the network. One of the objectives of this tool was that, later, the communication services could be configured using the GUI. The services implemented were QoS, *Virtual Private Network* (VPN) and a basic service of *Virtual Local Area Network* (VLAN), as well as the Hostname configuration of the equipment. One of the aims is to use the developed prototype to manage the referred network services. With this prototype we can, in just a few steps, configure QoS, VPN, VLAN or the hostname in a network. The hostname service would serve as proof of concept. After setting communication services, we have set up the configuration parameters of the service. To do this, we created a "skeleton service" to be implemented. In this "skeleton" there are several files, including the modelling of services, using the YANG. It is in the YANG files' that the fields, or parameters, are defined to be ordered for proper implementation of the services in the network. Figure 5 shows an example of part of a YANG file (*hostname.yang*) for implementing the hostname service, with the purpose of changing the hostname of the required device. This service, as mentioned previously, was created to demonstrate the implementation done and will be reflected in the tested network devices.

```
module hostname {
        namespace "http://com/example/hostname";
        prefix hostname;
        import tailf-ncs {
                prefix ncs;
        }
        container host {
                list hostname {
                        description "Configure
hostname";
                        key name;
                        uses ncs:service-data;
                ncs:servicepoint "hostname";
                leaf name{
                        type string;
                }
                  leaf device {
                        type leafref {
                          path
"/ncs:devices/ncs:device/ncs:name";
                        }
                  }
                  leaf changeto {
                        type string;
                  }
        }
      }
    }
```

Figure 5. YANG file for modelling a service: Hostname (*hostname.yang*)

In Figure 5 we can see the set parameters which will support the data to be filled in the NSO. On the YANG model we can see the name of the device whose hostname we want to change, and the new hostname we want to give it. If we run the command to create the Hostname service, it works, but only on the data storage in NSO CDB. After the change in YANG file, we must define the service mapping so that the command is executed and the service created. As for the mapping setting, this is nothing more than changing the template (*hostname.xml*) that is generated when we create the service in the NSO. In Figure 6 we present an example of Hostname service. the result may be the *template* shown next.

```
<config-template xmlns=http://tail-f.com/ns/config/1.0
servicepoint="hostname">
        <devices xmlns="http://tail-f.com/ns/ncs">
        <device>
                <name>{/device}</name>
                <config>
                        <hostname
xmlns="urn:ios">{/changeto}</hostname>
                        <configuration
xmlns="http://xml.juniper.net/xnm/1.1/xnm">
                                <system>
                                        <host-
name>{/changeto}</host-name>
                                </system>
                        </configuration>
                </config>
                </device>
        </devices>
</config-template>
```

Figure 6. Hostname service's *template* (*hostname.xml)*

In Figure 6 we can also note that the *template* already follows the hostname configuration, either to a Cisco *router*, identified by your operating system (IOS) or to the Juniper *router*, identified by your operating system (JunOS).

In Figure 7 we present a command that is an example of the Hostname service configuration and that may be used for practical implementation of changing of a device hostname, in this case, the *router p0*.

```
admin-ncs(config)# host hostname troca device p0
changeto p0cisco
admin-ncs(config)# commit
```

Figure 7. Example of command for Hostname service creation in NSO

After explaining the NSO, we will explain the creation of the GUI process that, for the network manager, is the only part that will be used for service management, after the network and the service are created, naturally.

3) Development tool

The final stage resulted in the development of a graphical interface where the user is expected to interact most of the time with regard to the service management part. The graphical interface was created in WordPress and is very simple. It is important to note that the main purpose was not the implementation of a high-level *web interface*, but the development of a solution that can serve as a stage prior to the configuration of the network and production service. We tried to create a simple and functional interface to make its use as easy as possible. There are more graphic tools with the function of network configuration, but most of them have many concepts which may not be necessary to those who will manage a network and its services [21]. The Cisco NSO technology is not very used yet but it is property of a big network company so it has potential. We have not found any related work with it, so to the best of our knowledge, our work is the first of its kind.

The implementation of the *WebUI* is divided into two parts: the visible (*front-end*) and non-visible (*back-end*), which are running the most important process. The *front-end* is very simple and it is based mainly on buttons and filling out forms. The *back-end* is where the data, that was previously filled in by the user forms, is read. In the *back-end* of the tool we have done the proper implementation to interpret and process everything the user sees. This reading follows the sequence shown in Figure 8.

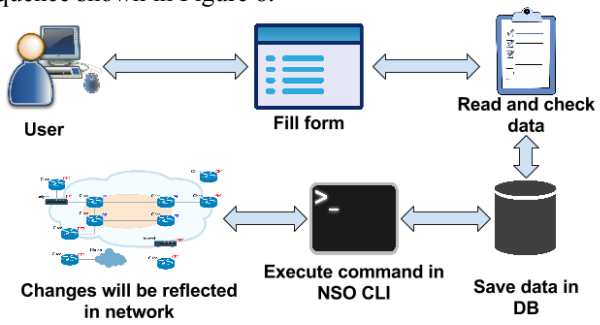

Figure 8. Process execution sequence runs in back-end in graphic interface

In what concerns the database, it is very simple and it is used mainly to synchronize the data to be presented in the form with the data on the NSO. The most important command, through which the connection between the GUI and terminal NSO is made, is shown in Figure 9.

```
$ /home/tail-f/ncs_new/bin/ncs_cli -C -u admin
```

Figure 9. Access command terminal of NSO

Running a *script* with this command is reflected in NSO terminal and later, in the existing network. The communication mode between the prototype and the NSO was the NSO [NCS] CLI Scripts [13], since it was the simplest and quickest way of implementing what we intended to test. Our NSO CLI Script is a solution available by NSO technology itself, thus it is a valid option to be used. There were other communication modes like the Python, REST and Java, depending on the type of solution to the management of network we have or we intend to develop.

To conclude the chapter, we present an example test of the entire process carried out.

B. Test of tool operation

On the graphical interface, the NSO checks the data after the user fills out a form for the hostname change. The form is shown in Figure 10.

 ISBN: 978-1-61208-482-4

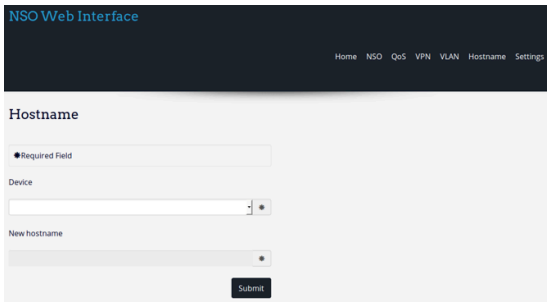

Figure 10. Hostname form, part of the graphic tool developed.

The parameters are validated after they are inserted. Only after their insertion will the commands be executed in the NSO terminal, the data is stored in the CDB and the mapping definition is made. This definition is reflected in the *template* result in the XML file, previously shown in Figure 6. Finally, the NED interprets the received data. The command is executed on the machine and the result is successful, as shown in Figure 11.

```
CE2#
CE2#
*Mar  3 15:49:08.208: %SYS-5-CONFIG_I: Co
(192.168.200.2)
CE2Cisco#
CE2Cisco#
CE2Cisco#sh run
CE2Cisco#show runn
CE2Cisco#show running-config
Building configuration...

Current configuration : 1288 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname CE2Cisco
!
```

Figure 11. Execution of commands sequence in back-end. Transparent process for the user.

All network services were implemented on the prototype. We did not develop all template services, because this work is expected to be done/developed by network or device manufacturers. Although we only present the test for hostname service, for proof of concept of the prototype tool, the results of testing QoS services will also be successful in Alcatel router. The changes were confirmed in this specific router.

We conclude the presentation of the implementation and of the demonstration of this tool execution.

V. Conclusion and Future Work

We proposed and implemented an *Open Source* tool that can be used to manage a network, and especially its services before they are put into production. Using the concept of SDN, the management can be done either in a real network or in a virtual one, whether it already exists or it is created from scratch. Its simple use allows the users to spend less time in the configuration and creation of services and, at the same time, it can be used to optimize both the network and the creation of new services. In practice, the process is simple: add a tool to a network and that tool is ready to be used. The configuration of the equipments, as it is done nowadays, will be maintained, but it will use a graphic tool so that this process becomes more simplistic and abstract to the user.

As future work, we can suggest the implementation of new services and the consolidation of this tool through a more optimized prototype. It would be an advantage to present this prototype to managers or network administrators, who work in this area daily, in order to improve this tool.

References

[1] Cisco Systems, Inc, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019”, 2015.

[2] HP Enterprise Business, “Why SDN… Software-defined Networking?”, 2014. Available from: <https://goo.gl/kfclyH>. Accessed on: December 05, 2015.

[3] Open Networking Foundation, “Software-Defined Networking (SDN) Definition. Open Networking Foundation”. Available from: <https://goo.gl/hMOCuy>. Accessed on: January 10, 2016.

[4] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, Proceedings of the IEEE (Volume:103 , Issue: 1), January 2015.

[5] Y. Jarraya, “A Survey and a Layered Taxonomy of Software-Defined Networking”, IEEE Communications Surveys & Tutorials (Volume:16 , Issue: 4), April 2014.

[6] Nuage Networks, “Products - Nuage Networks”. Nuage Networks. Available from: <http://www.nuagenetworks.net/products/>. Accessed on: December 15, 2014.

[7] Nuage Networks. Virtualized Services Platform, “Nuage Networks VSP Data Sheet”, June 2014. Available from: <http://goo.gl/Qj4nqB>. Accessed on: December 15, 2014.

[8] Cisco Systems, Inc., Tail-F Systems, “Cisco Network Service Orchestrator (NSO) enabled by Tail-f”. Available from: <https://goo.gl/Oy1BKH>. Accessed on: December 22, 2015.

[9] HP Enterprise Business. “Leverage SDN: Create consumable, programmable, and scalable cloud networks”, 2015, pp. 17.

[10] Nuage Networks, “Arista and Nuage Networks: Building Cloud Datacenters with OpenStack”, Dec. 01, 2015. Available from: <http://goo.gl/zJ4juN>. Accessed on: January 07, 2016.

[11] N. McKeown, G. Parulkar, T. Anderson, L. Peterson, H. Balakrishnan, J. Rexford, S. Shenker and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication Review, Volume 38 Issue 2, April 2008, pp. 69-74, doi: 10.1145/1355734.1355746.

[12] I. M. Kultan and Nuage Networks, “Virtualized Services Platform (VSP) & Network Services (VNS)”. Vienna, Austria, pp. 16. 2015.

[13] Cisco Systems, Inc, “Tail-f Network Control System 3.3 Getting Started Guide”, 2014, pp. 1; 3; 51-52; 59.

[14] Cisco Systems, Inc., Tail-F Systems, “What is YANG?” Available from: <http://www.tailf.com/education/what-is-yang/>. Accessed on: November 25, 2014.

 ISBN: 978-1-61208-482-4

[15] M. Rouse, “What is XML (Extensible Markup Language)?” TechTarget, Dec. 2014. Available from: <http://goo.gl/v65bZi>. Accessed on: January 10, 2016.

[16] R. Enns, M. Bjorklund, J. Schoenwaelder and A. Bierman, “RFC 6241 – NETCONF Configuration Protocol”, Jun. 2011. Available from: <https://tools.ietf.org/html/rfc6241>. Accessed on: December 31, 2015.

[17] Juniper Networks, Inc, “Junos OS NETCONF XML Management Protocol Developer Guide”, pp. 3. 2015.

[18] W. Stallings, “Software-Defined Networks and OpenFlow”, The Internet Protocol Journal, March 2013.

[19] J. J. Jensen, “Multi-Vendor Service Orchestration & Network automation for today’s networks”, 2016.

[20] GNS3 Technologies, Inc, “What is GNS3?”, 2016. Available from: <https://www.gns3.com/software>. Accessed on: February 21, 2016.

[21] L. D. Vecchio “GUI for Netfloc – An OpenSource SDK for SDN”, January 29, 2016.

 ISBN: 978-1-61208-482-4