Open Source Tool for Networks Management
Communication

Nuno Tiago Louro Simões¹ and Carlos Manuel da Silva Rabadão²
¹. School of Technology and Management, Polytechnic Institute of Leiria, Leiria, Portugal
². Research Center for Informatics and Communications, Polytechnic Institute of Leiria, Leiria, Portugal

Abstract: Considering the complexity of the networks and the need for telecommunication operators (Telco) to present innovations to their clients, the quickest possible, one of the solutions for this complexity could be to centralize its configuration. Thus the SDN (software-defined networking) 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.

1. 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 IT (information technology), telecom operators need to introduce new features to capture the customer’s attention. The time-to-market’s factor is also very important. The companies want to present novelties the quickest way possible. 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, also, 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 (software-defined networking), 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 2. In this section we will present some SDN solutions existing in the market, as well as some of the technologies used. In Section 3, we will present the proposed architecture that supports this work. The architecture contributes to the mitigation of previously presented problems. In Section 4, we will explain the implementation of our proposal. Finally, in the last section, we will present the conclusions and suggestions to work.

2. 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.

2.1 Background

Bearing in mind the evolution of networks, there has been an initial divergence between network companies which defended the virtualization and those who produced equipment. As time passed, there has been a convergence of two tendencies: Telco Network Evolution and Telco IT Evolution, as it can be observed in Fig. 1 [3].

![Evolution of SDN/NFV, according to Telco.](image1)

Fig. 1 Evolution of SDN/NFV, according to Telco.

Within the topic of programmable networks, a main solution has come up, with the name of SDN and a “twin” solution [4] called Network Functions Virtualization.

According to the Open Networking Foundation, the SDN is the physical separation of the control plane and the forwarding plane of the network [5]. With SDN concept, the networks will be configured and managed in a centralized way [6], 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 [7]. Thus, it simplifies the understanding of the network, which means that operators can do their job more quickly and easily, according to the time-to-market’s factor. Consequently, the operators may have good financial profits, which is an advantage.

Before presenting some SDN solutions, we can observe, on Fig. 2, a generic scheme of SDN.

As it can be noticed, SDN can be divided in 3 parts: Application Plane, Control Plane and Data Plane. Next each part is going to be resumed [8]:

![Generic scheme of SDN/NFV.](image2)

Fig. 2 Generic scheme of SDN/NFV.
• Application plane: in this plane we can refer some net apps such as, for instance, orchestration applications. These applications will communicate with the network and take care of its performance;
• Control Plane: the aim of this plane is to implement all the coordination protocols needed to the good functioning of Data Plane;
• Data Plane: in this plane the packages headings are analyzed and the packages received are sent to their final destination.

This separation makes it possible to diminish the complexity, increasing the intelligence and turning the network more programmable, dynamic and efficient. It also enables the applications to deal with a single network device, not having to worry with further functioning details [9].

In order to connect the Control Plane to the Data Plane, there is a communication protocol directly attached to SDN architecture and that bounds the two functions. In most cases, this communication protocol is OpenFlow.

The three main reasons to use SDN domains are [9]:
• Scalability: possibility of adding controls in the network, as it is getting wider;
• Privacy: using several privacy policies in different SDN domains;
• Incremental development: possibility of separating the network in sections, thus, having the chance to create a new tests network that does not disturb the main/production network.

Now we will present some of the existing SDN solutions: VSC (virtualized services controller), by the internal company of Alcatel-Lucent, the Nuage networks [10, 11], NCS (network control system) [12] by Tail-f, currently owned by Cisco portfolio, and NSX by VMware.

2.2 SDN Solutions

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

2.2.1 VSC (Virtualized Services Controller)
VSC, based on Alcatel-Lucent Service Router OS [13], is the SDN solution control panel of Nuage Networks and the most powerful SDN controller in the industry [11, 14]. 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), QoS (quality of service) or security rules. In the VSC, the connection between the controller and the network routing is established through the communication protocol—OpenFlow [15]. 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 [16].

2.2.2 NCS (Network Control System)
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 “NSO (Cisco Network Service Orchestrator) enabled by Tail-f” [12]. 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 [17]. 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 composed by three parts: implementation, monitoring and infrastructure/network equipment. This structure is more or less predictable given the SDN architecture.

2.2.3 VMware NSX
Even though in its site [18] there is not any reference to SDN, the association of NSX to SDN is consensual in several sites on the internet [19-23].
NSX solution of VMware is a safe platform of network virtualization specific to data centers or SDDC (Software-Defined Data Centre)\(^1\), and it has appeared after the acquisition of Nicira Company by VMware, Inc. [23].

Just like in any other SDN solutions, the creation of NSX has emerged due to the complexity in which the networks have turned and, according to the introductory video NSX \(^1\), the virtualization of equipment in the network makes the configuration time shorter. In this solution, the network management is done through some applications and, in order not to make any physical changes in data centers, the best procedure is to put a software layer on top of the existing hardware.

In case of any failures in the network, the mistakes presented will be detailed and concise, so that they can help the people responsible in a faster and easier way.

The low OpEx and CapEx, the time-to-market and the network security are some of the advantages of NSX usage\(^2\).

2.3 Used Technologies

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, XML (eXtensible Markup Language), NETCONF (Network Configuration Protocol), OpenFlow and OpenStack.

2.3.1 YANG

The YANG is a data modeling language used for a data state configuration model. This language is used by the network configuration protocol—NETCONF—and is published in the RFC (request for comments) 6020 of September 2010. The YANG is related to the content and operations in layers of NETCONF [24].

2.3.2 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 has similarities to the HTML (Hypertext Markup Language). We can consider that the construction of XML is done by blocks which are identified by tags [25].

2.3.3 NETCONF

The NETCONF is generically used to make the management of network devices configuration and it is based on the encoding in XML [26]. This protocol defines basic operations that are equivalent to commands to be executed from the CLI (Command-Line Interface). As in XML, NETCONF also uses tags. One of the manufacturers that uses NETCONF on its devices is Juniper Networks [27].

2.3.4 OpenFlow

OpenFlow is a communication protocol that enables its users to develop and test services. It allows direct access and manipulation of the network devices conveyance plan, such as switches and routers, being them physical or virtual. The OpenFlow is implemented both on the devices of the network infrastructure and on the SDN software control.

2.3.5 OpenStack

OpenStack is an operating system cloud that makes it possible to control large quantities of computers, storage and network resources through a data center. This control is made through a dashboard presented on a web interface.

In the next section, the architecture used on the suggested solution will be presented.

3. 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 Fig. 3 that are: Application Plane, Control Plane and Data Plane.
Fig. 3 Logical structure of SDN.

If we analyze, this logical structure is very similar to the one presented by William Stallings on his article [9]. Next we will explain each layer mentioned above:

- 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 analyse 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 Fig. 4 is quite simple, as it is divided into three layers: user, orchestration of the network and, finally, the network itself.

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.

3.1 Architecture Used in the Implementation

A more detailed architecture proposed for the development of this work is shown in Fig. 5.

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 Fig. 5, this communication can 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.

Some important steps to give on the application of this architecture are:

- Filling the NSO interface: after the user fills the NSO application form and confirm this procedure, the interface communicates with NSO. The implemented interface verifies some basic data such as the validation of IPv4;

- Communicating with NSO: the interface communicates with NSO, sending it the data received by the interface and, consequently, introduced by the user. All the system intelligent process is in the NSO. It is in NSO that all the data to send are made and it is, also, the Network Service Orchestrator that sends the information into the network. This means, in this step the configurations should already be outlined and properly configured, and converted if it is necessary, to be executed and interpreted by the network equipment, such as the routers.

It is in the stage of communication between the NSO and the network that the entire process for the correct operation is taken. The NSO is divided into four parts (three layers and a part relating to data storage) [17]:

- 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 directly connected devices. The services should be defined, and implemented, 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;
- CDB (Configuration Database): 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;
- NED (Network Element Drivers): 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 is the responsibility of the NED or the OpenFlow controllers, as we can see in Ref. [28]. 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 the SSH
(secure shell) communication protocol.

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

4. Prototype

The implementation of this tool is based on the architecture presented in Section 3. 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.

4.1 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 GUI (graphical user interface) 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.

4.1.1 Scenario/Network Topology

Initially a virtual Linux Ubuntu machine was created to run the 14.04 version. In this machine a network was developed on a network simulation software GNS3 [29], shown in Fig. 6, 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 OVA (open virtual application) image, more specifically a 12.1X47-D15.4 version of JunOS vSRX. For the Alcatel-Lucent equipment, we have used the model “ALCATEL SR-OS 7750 TiMOS-B-12.0.R6”.

The only settings made in this equipment were addressing, routing, the OSPF (open shortest path first) in this case, and the communication protocol configuration used—SSH.

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

4.1.2 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. The first objective

![Network topology defined to test developed in GNS3.](image-url)
was this (implement and test some services on the network) and next (simplify this objective, i.e., configure the services using the GUI). The services implemented were QoS, VPN (virtual private network) and a basic service of VLAN (virtual local area network), 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 modeling 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. Fig. 7 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.

In Fig. 7 we can see the set parameters which will support the data to be filled in the NSO. On the YANG model we can note 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. On Fig. 8 we can observe the abstraction obtained from the network when executing commands.

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 Fig. 9 we present an example of Hostname service. The result may be the template shown next.

In Fig. 9 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). Although it is expectable that the big companies and producers create the templates, or files, XML according to what they want. These are only good to exemplify a possible creation of a new service, so we can conclude that they facilitate the process to a great extent.

---

Fig. 7  YANG file for modeling a service: Hostname (hostname.yang).

Fig. 8  Abstraction in the execution of commands.

Fig. 9  Hostname service’s template (hostname.xml).
In Fig. 10 we present a command that is an example of the Hostname service configuration and that may be used for practical implementation of changing a device’s hostname, in this case, the `router CE2`. 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.

4.1.3 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 [30]. 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 Fig. 11.

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 Fig. 12.

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 [17], since it was the simplest and quickest way of implementing what we intended to test.

The execution of the command on Fig. 12 will act on the existing network, as it takes to the execution of the command presented on Fig. 13, as it can be seen next.

As we can observe, the parameters of the application, from the application form, will be replaced in the command shown on Fig. 13. Next, on Fig. 14, we present an example of a script, so that it can be executed on NSO.

![Fig. 10 Example of Hostname’s command for Hostname service creation in NSO.](image)

![Fig. 11 Process execution sequence runs in back-end in graphic interface.](image)

![Fig. 12 Access command terminal of NSO.](image)

![Fig. 13 Access Command to be incremented on bash script.](image)
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.

4.2 Prototype Implementation

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 Fig. 15.

The parameters are validated after they are inserted. Only after their insertions the commands will be executed in the NSO terminal, the data are 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 Fig. 9. Finally, the NED interprets the received data. The command is executed on the machine and the result is successful, as shown in Fig. 16.

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 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.

5. 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 or in a virtual network, 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 equipment, 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. The advantage of this prototype is the development of a graphic interface which presents and
takes advantage of the management of network and services in an easier and more visual mode, instead of what happens in traditional networks. The presented tests fundament this advantage, since, until now, according to the recent research, there is not any tool similar to the suggested one, using the Cisco NSO solution.

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. Besides those that have already been mentioned, another future works could be:

- Testing the tool using a mixed scenario (with real and virtual equipment) and a scenario with real equipment;
- Using other similar solution to “Cisco Network Service Orchestrator enabled by Tail-f”, so that it may be possible to test the differences and verify eventual improvements between the use of this other solution and the proposed tool;
- Using another way to access communication between the proposed tool and the NSO, this is to say that, instead of using scripts in bash, we should use REST or Python, for instance;
- In case people want to keep going and to stimulate other network, another simulation tool could be used, in alternative to GNS3, to develop the scenario of tests, once the tool, with different software, could have other type of unpredictable answers or it may need other type of changes in the proposed alteration.

We conclude the paper related to SDN, about using this concept to manage communication services.

References


