Simulation of OpenFlow Scenarios based on NSDL and NS-3

Ramon R. Fontes
Bahia Federal Institute of Education, Science and Technology (IFBA)
Salvador, BA – Brazil
ramonfontes@ifba.edu.br

Paulo N. M. Sampaio
Salvador University (UNIFACS)
Salvador, BA - Brazil
pnms.funchal@gmail.com

Talita R. Pinheiro
Salvador University (UNIFACS)
Salvador, BA – Brazil
talitarochapinheiro@gmail.com

Eduardo M. D. Marques
Universityof Madeira (UMa)
Funchal Madeira, Portugal
emarques@uma.pt

Keywords

OpenFlow, NSDL, Software Defined Networks, Visual Network Description, NS-3.

ABSTRACT

The management of OpenFlow experiments can be a complex task considering the number and the degree of configuration of the resources involved. This task would be straightforward if the existing heterogeneous tools could be applied together in order to optimize the management of these networks. The work presented in this paper applies the NSDL Framework in order to provide a common environment for enabling interoperability among OpenFlow compliant tools. In particular, we take advantage of the interoperability provided by NSDL for the utilization of different network tools and libraries aiming to create and analyze Software Defined Networks focusing on OpenFlow scenarios. Therefore, this paper addresses the utilization of a graphical user interface, called Visual Network Description, for the authoring of OpenFlow scenarios, the simulation of these scenarios using NS-3, and the application of the NSDL Framework for the integration of these tools. In particular, the integration provided by NSDL will enable the utilization of independent tools as complementary tools, allowing a more complete management and analysis, and contributing to enrich the best practices in OpenFlow networks authoring and management.

1. INTRODUCTION

The packet-switched network infrastructure is currently composed of expensive, proprietary equipment whose basic architectures are conceived by the combination of dedicated circuits responsible for ensuring high performance to data processing. The lack of flexibility to the control of the equipment internal mechanisms associated with their high costs represents an obstacle to the evolution of architectures and to innovation.[1][2].

The OpenFlow technology has been conceived to support the software (remote) control of network equipments, enabling the concept of software-defined networks (SDNs). SDN is a solution where a considerable amount of logical decisions a network device has to take are delegated to external controllers implemented as software [2]. OpenFlow was proposed at University of Standford in order to meet the demand for validation of new network architectures and protocols within commercial equipments. OpenFlow defines a standard protocol to determine packet delivery actions within network devices as, for instance, switches, routers and wireless access points [3]. The rules and actions implemented in network hardware are under responsibility of an external element, so-called *controller*, which can be implemented in an ordinary server [4].

Currently, there are several tools to support the OpenFlow implementation, such as: Flowvisor [5], Mininet [6], ENVI [7], Beacon [8], Nox [9] and NS-3 [10]. Nevertheless, the existing OpenFlow tools are currently diverse and heterogeneous, some of them being either open-source or proprietary, with different number of functionalities. However, if these tools were able to be applied jointly in a coordinated way, they could provide a solid and helpful environment for optimizing the management of OpenFlow networks. Unfortunately, the internal data formats used by these tools are very distinct and, most of the times, incompatible. In order to cope with this issue, it is important the proposal of an integration framework in order to bring together the implementation efforts of these tools and to provide support for the interoperability among these heterogeneous network tools.

This paper addresses the utilization of the NSDL Framework to provide the interoperability among different network tools and, thus, optimize the network authoring and management [12]. In particular, the NSDL Framework is applied to provide the integrated authoring and simulation of OpenFlow scenarios based, respectively, on the tools *Visual Network Description (SDN version)* and NS-3.

The NS-3 simulator is applied in order to simulate the OpenFlow scenarios created within VND (SDN version). This paper introduces the proposed mapping procedure between the NSDL Framework internal data structure (XML) and NS-3 scenarios (C++).

This paper is organized as follows: Section 2 presents the NSDL Framework; Section 3 introduces the VND (SDN version) authoring tool; Section 4 presents the mapping process between NSDL and NS-3. Section 5 presents a case study and finally some conclusions are drawn in Section 6.

2. NETWORK SCENARIO DESCRIPTION LANGUAGE (NSDL)

2.1 NSDL Framework

The main goal of the NSDL Framework is to provide an integration approach for heterogeneous tools in order to optimize the network management tasks. This Framework is composed of three layers, as depicted in Figure 1.

The first layer on the top is responsible for the interaction with the user, and includes all the graphical user interface (GUI´s) tools. Some functionalities of this layer include the network objects characterization, network monitoring, topology construction, events scheduling definition and also the presentation of analysis/simulation results.

The middle layer is the core component of the Framework since it is responsible for providing the interoperability among all the network tools. This layer relies on a common representation among these tools, the *Network Scenario Description Language (NSDL)*. NSDL is an XML-based language that provides the description of a network topology, the properties of the network components and any other specific network domain/tools information such as localization, simulation, security, management and others.

The bottom layer refers to the libraries and/or applications responsible for the execution of a network simulation, analysis and/or virtualization. Note that the top and bottom layers are related to the existing (or still to be developed) network management tools.

In particular, the structure of NSDL language provides one important feature for the Framework since it presents a separate description of the network components (e.g. topology) from the description of the behavior of different network scenarios (e.g., context-oriented based on domain or tools characteristics). This separation represents a breakthrough for the management of network scenarios since (i) different perspectives can be described for the same network scenario and (ii) the network description can be reutilized within different network management tools. This feature provides complementary results and analysis for the scenario optimizing the network management tasks. The NSDL structure is further detailed in the next section.

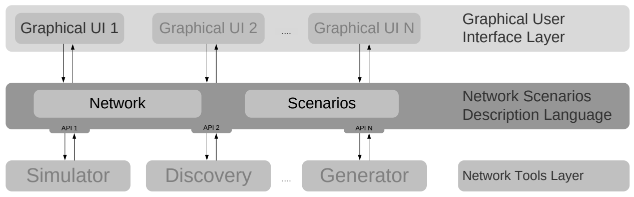


Figure 1. NSDL framework layered architecture.

2.2 NSDL Language

The main goal of NSDL is to provide a complete description of the network objects and their parameters and also a description of the several network scenarios throughout the network life cycle. Thus, the defined NSDL structure and parameters should also be representative enough to describe any type of network data. As an XML-based language, NSDL takes advantage of the richness and flexibility of this meta-language. NSDL is composed of a vocabulary and set of rules, both are able to support the description of wired and wireless data networks and the information concerning the context (or domain), where those networks are deployed or evaluated [12].

An NSDL representation is composed of two basic elements (as depicted in Figure 2): *Network* and *Scenarios*. The *Network* element contains the description of a network topology identifying its objects and its parameters. The *Scenarios* element may contain several descriptions, each one referring to a specific use, or context, to that respective network.
The *Network* element is composed of *Templates*, *Objects* and *Views*. The *Objects* element is the main component of the language since it contains the description of the network topology.

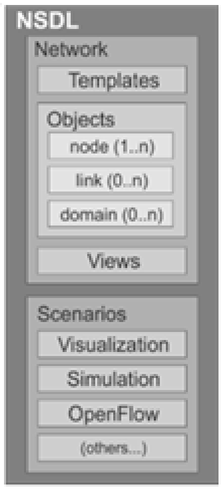


Figure 2.Structure of the NSDL Language.

This element is composed of *nodes*, *links* and *domains*. Inside the element *node* there are three other objects: *interfaces*, *protocols* and *applications*. Indeed, these six objects are representative enough to represent any current or future network element. It is also important to note that the language can be extended in order to describe new objects or network domains.

Some other important, but not mandatory elements are *Templates* and *Views*. The *Templates* element is important to simplify the description of similar objects, allowing the creation of a template object and the instantiation of it several times in the objects section, inheriting all properties. The *Views* element is a mechanism to group network objects and could be used to provide a grouped behavior or the characterization of a network segment and, most important, also being used by the scenarios as an object.

In the *Scenarios* element, three other elements are introduced: *Visualization, Simulation and OpenFlow*. The *Visualization* element provides additional information to enrich the network description, such as objects positioning in the GUIs and graphical data. All the parameters needed to implement a particular simulation over the network using a generic or particular simulation tool is defined in the *Simulation* element and the *OpenFlow* element provides information about OpenFlow configurations, such as *controllers* and *flowtables*.

The use of a single language description in several moments of a network life cycle can be very advantageous. If the users responsible for the network are familiar with that language, they can easily understand the current state of the network, thus management of the network can be optimized. Further integration capabilities can also be achieved since developers of a network tool are able to add import and export capabilities from and to NSDL, making their work inter-operable with other NSDL-compliant tools.

Next section introduces the graphical tool applied for the creation and edition of OpenFlow scenarios.

3. VND (SDN version)

The utilization of authoring and monitoring tools is useful in order to facilitate the network management during planning, observation and optimization of network operation. Furthermore, it provides the graphical visualization of all the connectivity and communication links (logical and physical) among the existing components. Nevertheless, since the authoring provides a static view of the network scenario, it is important to promote the integration of this tool with a simulation tool in order to provide the analysis and validation of the behavior of the respective scenario.

Visual Network Description (VND) (SDN version), presented in Figure 3, is a web-based graphical user interface for the authoring of generic network scenarios (also referred to as experiments) which allows the automatic generation of NSDL, and can be further applied for the simulation and analysis of these scenarios.

One of the main advantages of this tool, beyond its simple and intuitive interface, is the flexibility for customizing objects and their parameters for further creation of network scenarios. Another relevant feature of VND (SDN version) is its ability to automatic exporting to NSDL the current network scenario.

VND (SDN version) was conceived to be a versatile and adaptable tool, in opposite to most of the current available authoring tools. Therefore, it allows users to customize their authoring process according to their needs. The customization is carried out using an input XML-based file for the description of the library of objects. This file allows the description of three main groups of elements and their respective attributes that can be applied to create network scenarios using VND (SDN version): (1) *Instances*, which represent objects or groups of objects that can occur one or more time (such as *interface* or *protocol*); (2) *Objects*, which describe any type of component that can be used in a scenario and its parameters, and; (3) *Links*, which describe the connections between the objects.

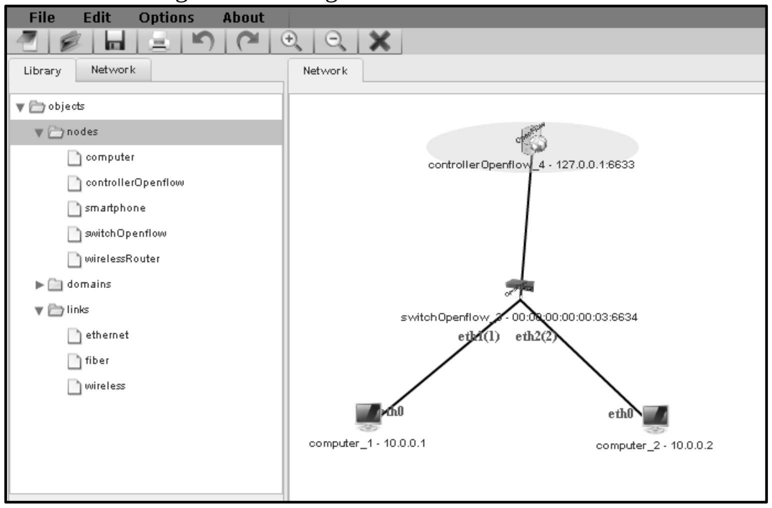
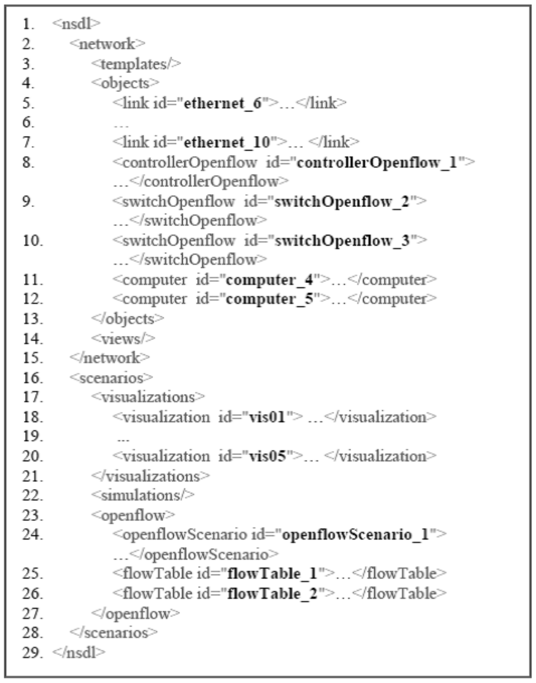


Figure 3.Creation of a network scenario with VND (SDN version).

VND (SDN version) allows the creation and configuration of an OpenFlow experiment. For this purpose, the user can apply either different pre-defined objects, such as nodes, links, computers, etc. or still can apply some OpenFlow objects such as *switchOpenFlow* (which redefines a switch element with OpenFlow functionalities) and *controllerOpenFlow* (which specifies a host with controller characteristics along with the Flow Table specification). As the user builds the topology of his scenario, he is also able to configure the respective parameters of all the objects applied.
Once the user has created his network scenario using VND (SDN version), he is also able to automatically export it to the NSDL format.
A general view of the respective NSDL file generated for the OpenFlow scenario is depicted in Figure 4.In this Figure it is possible to observe the objects included in the topology (*Network* element) as well as the contextual information for visualization and OpenFlow experiment (*Scenarios* element).



```
<nsdl>
  <network>
    <templates/>
    <objects>
      <link id="ethernet_6">…</link>
      …
      <link id="ethernet_10">… </link>
      <controllerOpenflow  id="controllerOpenflow_1">
      …</controllerOpenflow>
      <switchOpenflow  id="switchOpenflow_2">
      …</switchOpenflow>
      <switchOpenflow  id="switchOpenflow_3">
      …</switchOpenflow>
      <computer  id="computer_4">…</computer>
      <computer  id="computer_5">…</computer>
    </objects>
    <views/>
  </network>
  <scenarios>
    <visualizations>
      <visualization  id="vis01"> …</visualization>
       ...
      <visualization  id="vis05">… </visualization>
    </visualizations>
    <simulations/>
    <openflow>
      <openflowScenario id="openflowScenario_1">
      …</openflowScenario>
      <flowTable id="flowTable_1">…</flowTable>
      <flowTable id="flowTable_2">…</flowTable>
    </openflow>
  </scenarios>
</nsdl>
```

Figure 4. NSDL overview.

With the respective NSDL file describing the OpenFlow scenario (Figure 4) it is possible to apply any NSDL-compliant tool in order to proceed with the execution, simulation and analysis of the created experiment. In this case the utilization ofNS-3 was proposed to illustrate the integration provided by NSDL. Next section discusses the mapping process between NSDL and NS-3 (C++) descriptions for the created OpenFlow scenario.

4. MAPPING NSDL INTO NS-3

This section illustrates how the NS-3 description (C++) is generated from the respective NSDL file for the OpenFlow scenario.

NS-3 has been adopted for the simulation of OpenFlow scenarios since it is a flexible Open Source application. The network scenarios described in NS-3 are characterized by their simplicity and objectivity since they are written in an object-oriented language.

In Figure 5it is possible to observe the proposed steps for the translation between NSDL and NS-3. Initially, the NSDL file is uploaded on the NSDL project webpage [15]. After that, this file will be analyzed for lexical and semantic validation. Finally, the generation of the respective C++ code is carried out and then the output file can be download.

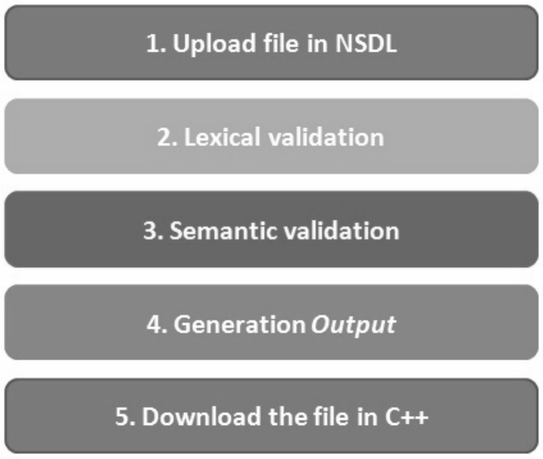


Figure5. Mapping NSDL/NS-3.

In order to execute these steps (depicted in Figure 5), it was necessary to create a script to enable the automatic translation between NSDL and C++ for the respective OpenFlow scenario.

In the next section, some equivalences between NSDL and C++ for an OpenFlow scenario are analyzed, which contributes to the automatic mapping between these representations.

4.1 Equivalences between NSDL and C++

In order to propose a script for providing the automatic mapping between NSDL and C++, it is important to understand the equivalences between the components of an OpenFlow scenario described in NSDL and NS-3 (C++).

This equivalence can be illustrated by the utilization of a simple scenario, as depicted in Figure 6. This scenario is composed of two hosts, a *switch OpenFlow* and a *Controller*.

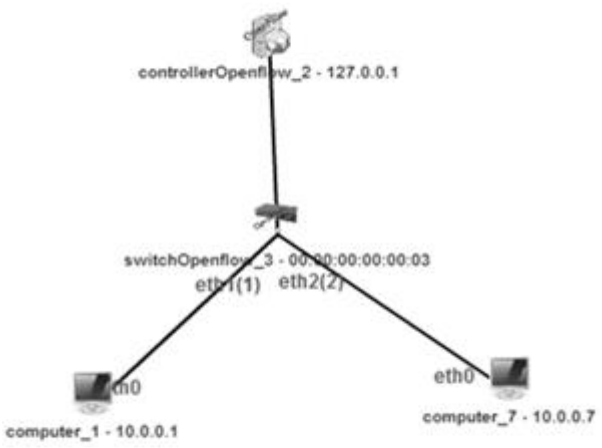
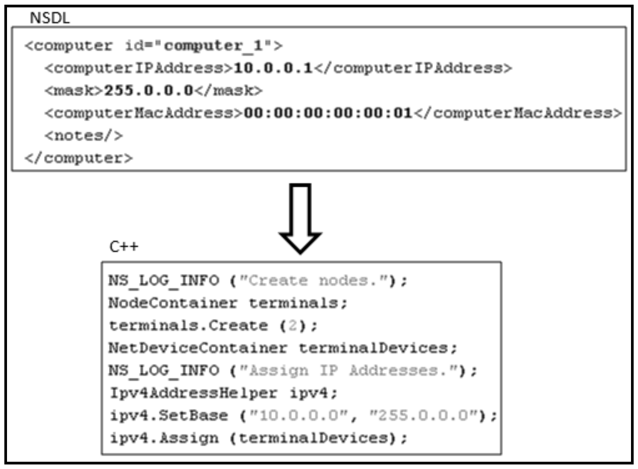

controllerOpenflow_2 - 127.0.0.1
switchOpenflow_3 - 00:00:00:00:00:03
eth1(1)
eth2(2)
eth0
eth0
computer_1 - 10.0.0.1
computer_7 - 10.0.0.7

Figure6. Topology Scenario OpenFlow.

The topology depicted in Figure 6 was conceived using the VND (SDN version) tool which in turn automatically generated the respective description of the scenario in NSDL. In order to propose this equivalence, we also have the scenario in C++. In the sequence, we verify some of these equivalences and particularities.


NSDL

```
<computer id="computer_1">
  <computerIPAddress>10.0.0.1</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
  <notes/>
</computer>
```

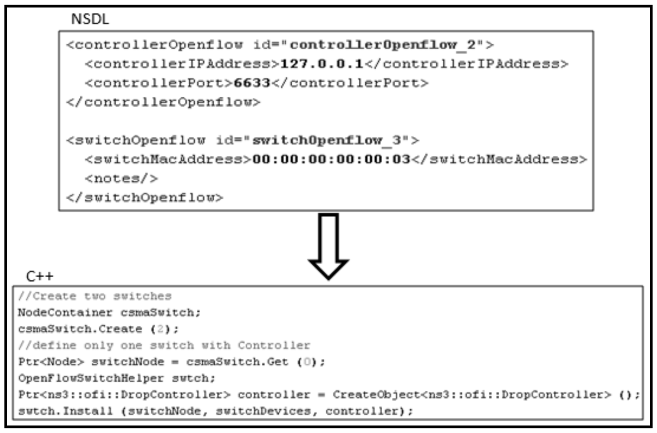
C++

```
NS_LOG_INFO ("Create nodes.");
NodeContainer terminals;
terminals.Create (2);
NetDeviceContainer terminalDevices;
NS_LOG_INFO ("Assign IP Addresses.");
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.0.0.0", "255.0.0.0");
ipv4.Assign (terminalDevices);
```

Figure 7. Description of Hosts in NSDL and C++.

Figure 7 illustrates the description of hosts (computers) either in NSDL and C++.As we can verify in the respective NSDL code for the description of a host, during the creation of a "Computer" object, some parameters are defined, such as the IP, mask and MAC addresses. The respective C++ code can only be created after this addressing has been done.

Since NS-3 does not support the description of a Controller, the NS-3 (C++) description for a switch can also be applied to a Controller. Therefore, the C++ description of a switch will provide an internal controller to manage the network. Figure 8 depicts the description of a switch.


NSDL

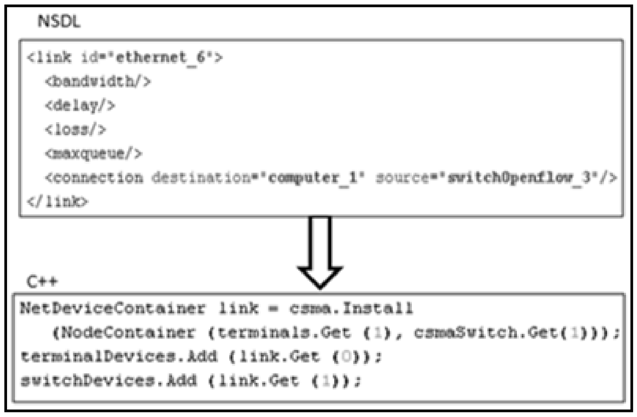
```
<controllerOpenflow id="controllerOpenflow_2">
  <controllerIPAddress>127.0.0.1</controllerIPAddress>
  <controllerPort>6633</controllerPort>
</controllerOpenflow>

<switchOpenflow id="switchOpenflow_3">
  <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
  <notes/>
</switchOpenflow>
```

C++

```
//Create two switches
NodeContainer csmaSwitch;
csmaSwitch.Create (2);
//define only one switch with Controller
Ptr<Node> switchNode = csmaSwitch.Get (0);
OpenFlowSwitchHelper swtch;
Ptr<ns3::ofi::DropController> controller = CreateObject<ns3::ofi::DropController> ();
swtch.Install (switchNode, switchDevices, controller);
```

Figure 8. Description of a Switch and Controller OpenFlowin NSDL and C++.

Figur9 illustrates the description of a link between network objects in NSDL and NS-3 (C++).As it is possible to verify either in NSDL and C++, the connection of two specific objects is established by the assignment of some parameters such as bandwidth, delay, loss and max length of a queue.


NSDL

```
<link id="ethernet_6">
  <bandwidth/>
  <delay/>
  <loss/>
  <maxqueue/>
  <connection destination="computer_1" source="switchOpenflow_3"/>
</link>
```

C++

```
NetDeviceContainer link = csma.Install
   (NodeContainer (terminals.Get (1), csmaSwitch.Get(1)));
terminalDevices.Add (link.Get (0));
switchDevices.Add (link.Get (1));
```

Figure 9. Description of a Link in NSDL and C++.

As illustrated in Figures 7, 8 and 9, a similar equivalence has been proposed for the description of all the components of an OpenFlow scenario, such as: Computer, OpenFlow switch, controller, links, etc. As already highlighted, the proposal of the equivalence for these components served as a basis for the development of the script for the automatic mapping between NSDL and NS-3.

Next section presents further details about the script developed.

4.2 Script for Automatic Generation

The development of the script provides the integration of the NS-3 simulator with the framework NSDL. Using the proposed script, it is possible to load an NSDL scenario and download a C++ scenario.

As for the NSDL scenarios lexical and semantic validation, Schemas were created in order to enable checking all tags.

After the lexical and semantic validations, an XSLT file is applied to the NSDL code to generate C++ code. The XSLT is an extensible stylesheet language transformation and is responsible for transforming the NSDL file into C++.

The structure of the script is based on a common C++ code structure for the simulation of OpenFlow scenarios, as illustrated in [16].

The main steps implemented in the proposed script for the generation of the respective C++ code are described as follows:

1 - Inclusion of libraries for NS-3;

2 - Definition of global variables (Realtime and CheckSum for example);

3 - Instantiating a helper for network scenarios (OpenFlowHelper);

4 - Definition of typical ports for traffic (UDPport = 9, Tcpport = 50000, Controllerport = 6633);

5 - Instantiation of objects and its specifications (Computer, OpenFlow Switch and Controller);

6 - Installation of Internet and the global routing (AODV, OLSR, DSDV, etc.);

7 - Creating Link(s) and the respective parameterization (EthernetChannel);

8 - Defining Interfaces (OpenFlow switch device);

9 - Addressing the nodes;

10 - Definition of Applications and their parameters;

11 - Definition of PacketSink and its events (Start time, Stop time);

12 - Configuration of the Simulation main attributes (Simulation Start time, Stop team simulator destructor, etc.);

13 - Visualization Setup (PyViz);

14 - Creating Output Files (trace files (.Tr) and Pcap files (.pcap).

Next section discusses a case study with the application of the proposed methodology.

5. CASE STUDY

The main goal of this case study is to illustrate the integration of VND (SDN version) and NS-3 for the creation and execution of OpenFlow scenarios using the NSDL Framework.

As already referred in this paper, the VND (SDN version) is applied for the authoring of OpenFlow scenarios. After the creation of this scenario, it is possible to automatically generate a respective NSDL description. In order to provide the integration with other OpenFlow compliant tools and, in this case, with the simulator NS-3, we have proposed the development of a script for the automatic mapping between NSDL and NS-3 (C++).

Figure 10 presents the life cycle of an OpenFlow scenario depicted within the NSDL Framework, starting with its creation using VND (SDN version) as graphical tool applied in the upper layer of the framework NSDL, the subsequent automatic generation of the respective NSDL description (middle layer), and the final mapping to the respective NS-3 description (bottom layer).

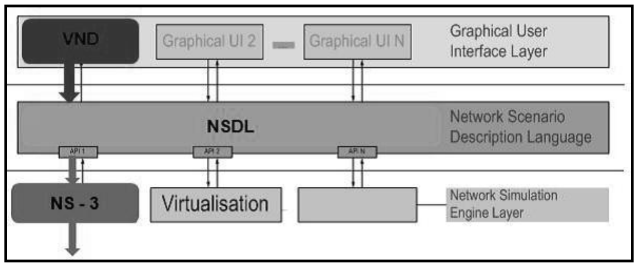

VND
Graphical UI 2
Graphical UI N
Graphical User Interface Layer
NSDL
Network Scenario Description Language
API 1
API 2
API N
NS - 3
Virtualisation
Network Simulation Engine Layer

Figure10. Case Study.

Some aspects are presented in the sequel illustrating the creation of topology within VND (SDN version), code generation in NSDL and finally, the implementation in NS-3 through the code in C++.

Initially, the topology of the scenario was created using VND (SDN version). During the authoring of the scenario it is necessary to apply the nodes that compose the scenario (hosts, OpenFlow Switches, Controller and links) and configure the respective attributes for each component. All the components of the scenario can be configured using some interactive windows and frames. The topology applied in this case study is the same as presented in Figures 3 and 6.

Once the scenario has been composed and configured, the description of the scenario can be exported to NSDL. The exporting functionality can be accessed using the main menu, as depicted in Figure 11.

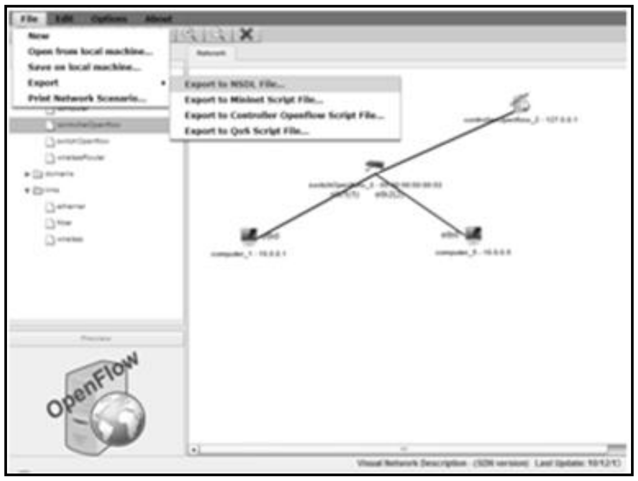
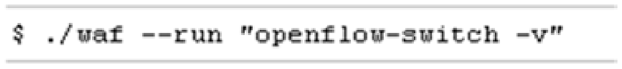

File
Edit
Options
About
New
Open from local machine...
Save on local machine...
Export
Print Network Scenario...
Export to NSDL File...
Export to Mininet Script File...
Export to Controller Openflow Script File...
Export to QoS Script File...
OpenFlow

Figure 11. Creating topology and exporting to NSDL file.

After the generation of the NSDL code, it is possible to map automatically this description to any target code available in order to integrate with other OpenFlow compliant or simulation tool. In this case, we can apply a developed script to provide the automatic mapping between NSDL and NS-3 (C++).

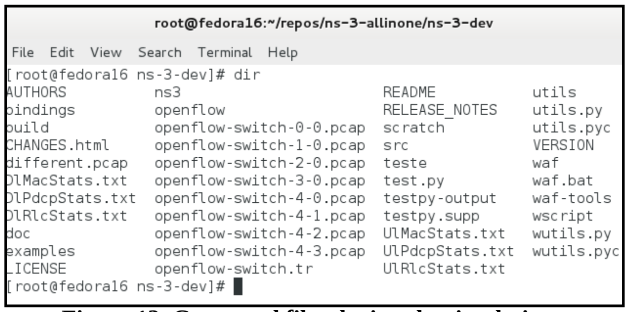
Once the C++ code for the respective scenario is generated, the simulation of the scenario can be executed using NS-3. For this purpose, the following command can be executed (Figure 12):



```
$ ./waf --run "openflow-switch -v"
```

Figure 12. Execution in NS-3.

During the execution, NS-3 generates a set of files (Figure 13) corresponding to the expected output of a simulation. One of these files are the *tracer* file and *pcaps* files with which it is possible to analyze the behavior of the network during the simulation.



```
root@fedora16:~/repos/ns-3-allinone/ns-3-dev
File  Edit  View  Search  Terminal  Help
[root@fedora16 ns-3-dev]# dir
AUTHORS          ns3                          README           utils
bindings         openflow                     RELEASE_NOTES    utils.py
build            openflow-switch-0-0.pcap     scratch          utils.pyc
CHANGES.html     openflow-switch-1-0.pcap     src              VERSION
different.pcap   openflow-switch-2-0.pcap     teste            waf
DlMacStats.txt   openflow-switch-3-0.pcap     test.py          waf.bat
DlPdcpStats.txt  openflow-switch-4-0.pcap     testpy-output    waf-tools
DlRlcStats.txt   openflow-switch-4-1.pcap     testpy.supp      wscript
doc              openflow-switch-4-2.pcap     UlMacStats.txt   wutils.py
examples         openflow-switch-4-3.pcap     UlPdcpStats.txt  wutils.pyc
LICENSE          openflow-switch.tr           UlRlcStats.txt
[root@fedora16 ns-3-dev]# █
```

Figure 13. Generated files during the simulation.

Also, based on the output files of an NS-3 simulation, it is possible to visualize the graphical animation for the corresponding scenario. In particular, Figure 14 illustrates the animation of the simulation for the scenario depicted in Figure 10. For the visualization of this scenario, the PyViz tool [17] has been applied.

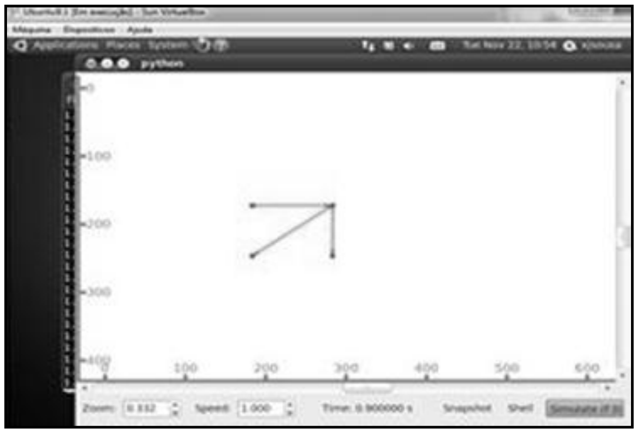


Figure14. Simulation on NS-3.

6. CONCLUSIONS

This paper discussed the viability to integrate heterogeneous OpenFlow compliant tools using the NSDL framework. In particular, this interoperability was illustrated with the utilization of a Graphical Authoring Tool called VND (SDN version) for the creation of an OpenFlow scenario and further validation with NS-3.

Due to the number of tasks associated with the configuration of Software Defined Networks, in particular, OpenFlow, the VND (SDN version) has demonstrated to be a useful and intuitive tool in order to provide a rapid prototyping of these scenarios. With the utilization of the NSDL Framework, it is possible to further validate and test these scenarios with different existing tools, in this case, with the use of the NS-3 simulator.

Indeed, the integration of heterogeneous tools provided by NSDL allows an in-depth study of OpenFlow networks since complementary perspectives can be achieved over the same scenario. As for future works, we aim at extending the integration of more complex OpenFlow scenarios created with VND to other OpenFlow compliant tools.

7. REFERENCES

[1] ANWER, M. B. et al. Switchblade: a platform for rapid deployment of network protocols on programmable hardware. In: SIGCOMM ’’10. Proceedings. New York, USA: ACM, 2010. p.183-194.

[2] Rothenberg, C.E. et al. OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. In: Cad. CPqD Tecnologia, Campinas, v. 7, n.1, p. 65-76, jul. 2010/jun. 2011.

[3] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008) “Openflow: Enabling Innovation in Campus Networks”, SIGCOMM CCR, 38:69–74.

[4] Sherwood, R. et al (2010) “Carving Research Slices Out of Your Production Networks with Openflow”, SIGCOMM Comput. Commun. Rev., 40:129–130.

[5] Flowvisor. Retrieved March 17, 2013, from https://github.com/OPENNETWORKINGLAB/flowvisor/wiki

[6] Mininet. An Instant Virtual Network on your Laptop (or other PC). Retrieved March 17, 2013, from http://openflow.org/mininet

[7] ENVI. An Extensible Network Visualization & Control Framework. Retrieved March 17, 2013, from http://www.openflow.org/wp/gui

[8] BEACON. Retrieved March 17, 2013, fromhttps://openflow.stanford.edu/display/Beacon/Home

[9] Nox. Retrieved March 17, 2013, from http://noxrepo.org

[10] Simulator NS-3, from http://nsnam.org

[11] NS-3 OpenFlow, “OpenFlow switch support” fromhttp://www.nsnam.org/docs/release/3.11/models/html/openflow-switch.html

[12] E. M. D. Marques and P. N. M. Sampaio, "Nsdl: an integration framework for the network modeling and simulation," *International Journal of Modeling and Optimization*, vol. 2, no. 3, pp. 304-308, June 2012.

[13] XML, 2012. eXtensible Markup Language (XML). http://www.w3.org/XML/.

[14] ONF. OpenFlow Switch Specification. Version 1.3.2 (Wire Protocol 0x04), 2013. Available at https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.2.pdf.

[15] Project NSDL. http://apus.uma.pt/nsdl/

[16] SOUSA, J. J. F., "Autoria e Simulação de Cenários de Redes em NS-3". Portugal, Set/2011.

[17] PyViz Toll. http://www.nsnam.org/wiki/index.php/PyViz