OF-NEDL: An OpenFlow Networking Experiment Description Language Based on XML*

Junxue Liang[1], Zhaowen Lin[1], and Yan Ma[1,2]

[1] Research Institute of Network Technology,
[2] Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia
Beijing University of Posts and Telecommunications
100876, Beijing, China
{liangjx,linzw,buptnic}@buptnet.edu.cn, mayan@bupt.edu.cn

Abstract. OpenFlow is a promising future Internet enabling technology and has been widely used in the network research community to evaluate new network protocols, applications, and architectures. However, most of these research activities or experimentations are lack of a uniformed description so can be repeated by other researchers. In this paper, we investigate the general model of an OpenFlow networking experiment and propose a language, OF-NEDL, which aspires to bridge this gap by providing a simple, comprehensive and extensible language for describing OpenFlow networking experiment. OF-NEDL allows the researcher to write a script to control every aspect of an OpenFlow networking experiment, including the hierarchical network topology description, the OpenFlow network devices configuration, the experiment software deployment, the experiment process control, monitoring and output collection. Our preliminary usage scenario shows that it has the ability to describe simple but extensible networking experiment, and we expect to refine considerably its design to make it more practical in the future work.

Keywords: Future Internet, OpenFlow, Network Experiment, XML, OF-NEDL.

1 Introduction

The Internet has grown so successfully that it plays a crucial role in today's society and business. It interconnects over a billion people, running a wide range of applications and is regarded as a great success by almost every measure. However, the Internet comes with important shortcomings because it originally not been designed for this wide range of usage scenarios. It can't handle new requisites such as end-to-end quality of service, security, and mobility. The network research community is generally more interested in fundamental questions about the structure of the Internet, including the design of new networking algorithms, protocols, and services, even construct the Internet completely new from scratch [1]. Of course, any such ideas must be rigorously tested and evaluated before being deployed in the actual Internet.

* Foundation Items: The National High Technology Research and Development Program of China (863 Program) (2011AA010704)

F.L. Wang et al. (Eds.): WISM 2012, LNCS 7529, pp. 686–697, 2012.

A network testbed is simply a collection of networking and systems resources that share a common infrastructure, and provides environment which allows researchers to develop, test, and evaluate new ideas. During recently years, many kinds of network testbed have been developed and some of them were widely used in network research community, such as PlanetLab [2] and Onelab [3]. However, most of these testbeds have only PC nodes as user configurable resources, and provide only application level networking experiment facility for the researchers. Hence, it is almost no practical way to experiment with new network protocols (e.g., new routing protocols, or alternatives to IP) in sufficiently realistic settings. To address the problem, Software-Defined Network (SDN) [4] is proposed and getting a lot of attention. SDN restructures network and exposes network APIs so that any software can configure the network as they want. Recently, OpenFlow [5] has been developing and is considered as a promising candidate technology to realize SDN. In brief, OpenFlow is based on a model that provides a physical separation of the network forwarding function and the network control function. The forwarding elements are dummy elements that have a generic shared forwarding table, which is accessed and configured by the independent control element called controller. In this situation, it is much easier to change or add new features to an OpenFlow enabled network testbed.

To start experiment on an OpenFlow network testbed, the first problem is how to describe the experiment, including the resources will be consumed, the control of experiment processes, the result or output of the experiment and so on. In a sense, the testbed is similar to the network simulation tools (NS2 etc.) and needs an experiment description language to tell the testbed system how to execute the experiment. For example, PlanetLab has implemented its experiment description language based on XML. However, these languages are lack of support to describe the OpenFlow network devices and the characteristics of OpenFlow enabled experiments. So we need to explore the principles and design a uniform and systematic framework to describe an OpenFlow networking experiment.

In this paper, we present the preliminary prototype of the OpenFlow Networking experiment Description Language (OF-NEDL), which intends to make it possible to write a script to control every aspect of an OpenFlow networking experiment, including the hierarchical network topology description, the OpenFlow network devices configuration, the experiment software deployment, the experiment process control, monitoring and output collection. This script file may be used by other researchers to repeat the experiment. The open question we are trying to answer is how to define a uniform abstract framework that can be used to describe arbitrary OpenFlow-based networking experiment, while keep it easy for the new user and extensible to design more complex experiments.

The remainder of this paper is organized as follows. Section 2 reviews some of the related works. Section 3 outlines the model of OpenFlow experiment and the Web-based experiment infrastructure. Section 4 presents the design principle and general architecture of OF-NEDL, and then demonstrates a simple usage scenario to evaluate it in section 5. Finally, section 6 concludes this paper and presents some future work.

2 Related Work

NS2[6] is a widely used network simulation tool in network research community. It is a discrete event simulator that has native support for common networking protocols and components. The simulation description language for NS2 is an extension of the Tcl (Tool Command Language). Using the various available NS2 commands, it can define network topology, configure traffic sources and sinks, program arbitrary actions into the configuration, collect statistics and invoke simulation.

NS3[7] is another popular network simulator with more advanced feature than NS2, but not an extension of NS2. Its simulation script language is C++ or optional Python, not the Tcl. Recently, the NS3 community are developing an XML-based language called NEDL (Ns3 Experiment Description Language) and NSTL (Ns3 Script Templating Language) [8], which provide a foundation for the construction of better interfaces between the user and the NS3 simulator.

Emulab [9] is a large network emulator, based on a set of computers which can be configured into various topologies through emulated network links. The Emulab control framework supports three different experimental environments: simulated, emulated, and wide area networks. It unifies all three environments under a common user interface, definitely, it uses an extend ns script written in Tcl to configure an experiment. For the researchers familiar with ns, it provides a graceful transition from simulation to emulation. The cost of this feature, however, is that the specification language and the relationship between virtualized resources is complex which makes it difficult to parse at the syntactic level.

The Open Network Laboratory (ONL) [10] is another emulation testbed that is similar in some respects to Emulab. ONL provides researcher with an user interface named Remote Laboratory Interface (RLI). The primary goal of the RLI is to make user interaction with testbed resources as easy as possible. In RLI, the most important abstraction is the resource type abstraction, which is represented in an XML file and consists of two types: the base description and the specialization description. The base description represents the physical device, while the specialization description represents one set of functionality supported by a particular type. It allows the testbed to include highly configurable and programmable resources that can be utilized in a variety of ways by the users.

Plush (PlanetLab user shell) [11] is a set of tools for automatic deployment and monitoring of applications on large-scale testbeds such as PlanetLab. Using Plush, the researcher can specify the experiment through an XML-RPC or a web interface and then remotely execute and control the experiment. In Plush, the experiment description or application specification consists of five different abstractions, that is, the process, barrier, workflow, component, and application blocks. The combinations of these blocks provide abstractions for managing resource discovery and acquisition, software distribution, and process execution in a variety of distributed environments. However, Plush focus solely on the application-level experiment in an overlay network, and is hard to extend to OpenFlow network.

Gush (GENI user shell) [12] is an experiment control system that aims to support software configuration and execution on many different types of resources. It leverages prior work with Plush, and is developed as part of the GENI (Global Environment for Network Innovations) project [13]. Gush accomplishes experiment control through an easily-adapted application specification language and resource.

Similar to Plush, the main goal of Gush project is used to simplify the tasks associated with application deployment and evaluation in distributed environments, but is more adaptable with other GENI control frameworks such as ORCA (Open Resource Control Architecture) [14] and ProtoGENI [15].

The cOntrol and Management Framework (OMF) [16] provide tools and services to support and optimize the usage of testbeds on a global scale, through systematic experiment description and execution, resource virtualization, and testbed federation. OMF is heavily focused on supporting repeatable experiments through OEDL [17], a domain-specific language based on Ruby fully capturing the experiment setup, its orchestration, and all relevant contexts.

Others, such as NEPI (Networking experiment Programming Interface) [18] and Expedient [19], proposed alternative frameworks to OMF to describe, execute and instrument networking experiments. NEPI proposes a framework based on a unified object model to describe a networking experiment which could subsequently be executed on different environments (e.g. simulations, emulations, and testbeds). Expedient is an ongoing effort to provide a testbed control framework that aggregates heterogeneous APIs to the underlying resources such as PlanetLab and Emulab, allowing for a user-friendly Web-based configuration and control of the experiment slices.

3 The OF-NEDL Design

3.1 Principles

In the previous section we investigated the model and the process of an OpenFlow experiment executing on the Web-based experiment infrastructure. Based on the discussion in the previous section, we now extract some general principles for an OpenFlow network experiment description framework.

Simple. It should be simple and easy to use, so that the user can concentrate his attention on the experiment software design itself but not the experiment description. The XML tag names and element structures should be easy enough to write and edit by hand as well as automatically generated by some other module (i.e., a front-end GUI). Keep it simple also means that it is easy to parse or handle by the Backend, so that the development complexity is to be reduced.

Comprehensive. It should be comprehensive and capture the abstract nature of the experiment, identify all aspects of the execution and environment needed to successfully deploy, control, and monitor the experiment. Basically, an OpenFlow network experiment description should include following components:

a description of the resources needed and network topology;
a description of the OpenFlow experiment configuration and deployment;
a description of the experiment control flow;
a description of the expected output.

Extensible. It should be extensible and provide explicitly defined procedures and system interfaces, making it easy to incorporate additional resources and technologies, even including those that do not exist today. In particular, it should be

designed to minimize the effort of adding new node and link types, and ideally it should be handled without the need to modify any of the experiment infrastructure software.

The main challenge OF-NEDL attempts to overcome is the trade-off between simplicity and extensibility, which means that it should be easy to understand and write, while providing flexible constructs that allow for more complex experiments design.

3.2 The Architecture

According to the principles mentioned above, we have defined an OpenFlow Networking Experiment Description Language (OF-NEDL) to describe every task of a typical OpenFlow networking experiment workflow. As depicted in Figure 1, OF-NEDL is hierarchical. Experiment is defined at the highest level, and it contains information, topology, deployment, control, output components.

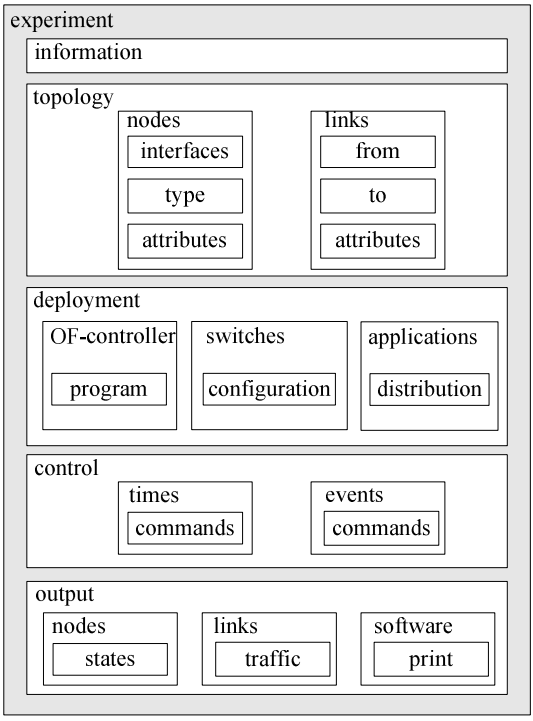

Fig. 1. General Structure of OF-NEDL

Information Component. Experiment information may include information about the user, group, experiment and global parameters. The most important is the setup of global parameters. These basic parameters, such as the start, duration, end times of experiment, the policy of error handle, will tell the Backend how to "treat" this experiment.

Topology Component. It is static representation of a group of nodes connected by links. Hence, it consists of two types of component, node and link. In communication networks, a node is either a redistribution equipment such as router, switch, hub, or a communication endpoint like a PC or sensor device. Each node has one or more interfaces to communicate with each other, and exhibit some attributes to materialize itself. To better illustrate the logical structure of the node component, we make use of UML class diagram. As show in Figure 2, a node has some general entries and attributes, which is similar to the Base Class in OOP (Object-Oriented Programming). Other special entries and attributes are stored in a Type structure whose type is one of the special devices (Router, Switch, KVM or openVZ virtual node, etc.) Another component, link, is the communication channel that connects two node through interfaces. To provide more configurable characteristics, the link is designed to have "from" and "to" endpoints to define a single direction communication. The attributes of link, such as delay, bandwidth, and loss-ratio are common for all kinds of physical channel, so there is no need to define "type" like node component.

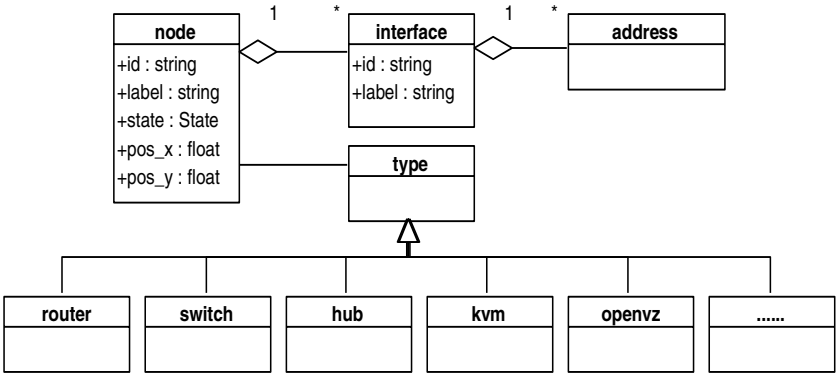

Fig. 2. UML Class Diagram for the Node Component

Deployment Component. The process of experiment deployment involves preparing the physical resources with the correct configuration and software. This typically includes copying and installing the OF program to OpenFlow controller, configuring the OF switches, copying and installing the software to all of the endhosts. According to the analysis of OpenFlow experiment in the subsection 3.2, the OF program and OF switch configuration are dedicated to themselves. Otherwise, the endhosts and software may have different combinations, so there may be two different ways to deploy the endhosts software, one is

```
SW_A: (EH_A, EH_B,... EH_X)
```

and another is

```
EH_A: (SW_A, SW_B,... SW_X)
```

where SW and EH represent the software and endhost. The main consideration is efficiency and compactness, especially when the number of endhosts and software increase to high, so we will provide the options to the user to decide which way to use.

Control Component. Experiment control includes start/stop the OF program and endhosts software, change the OF switches configuration, and issue commands on endhosts. All of these control actions may depend on times and events as following example:

```
AllReady: (ALL: CMD_A, CMD_B,... CMD_X)
   Wait_30S: (ALL_EXCEPT_EH_A•CMD_A)
```

The means of these expressions are literal, so it is easy for a novice user to get started. One problem is the control granularity provided. Fine granularity may provide user more control flexibility, but involve more complexity in the experiment infrastructure. We will refine the design details in the future work.

Output Component. The expected outputs in an experiment may vary widely due to the different goals and design. In principle, we categorize these outputs as node-related states, link-related traffic and software printed information. In general, the node-related states include the physical resources' states such as CPU, memory usage; the link-related traffic may contain the packets generated by the endhosts software and/or captured by the network interface, both of them may be filtered by special packet header; the software printed information is specific to the software and defined by the users.

3.3 XML Schema Samples

In this subsection we illustrate how we have translated the structure and class diagrams described in the previous subsection in an XML format that we describe by means of XML Schemas.

```
<xs:schema version="0.1">
  <xs:include schemaLocation="Information.xsd"/>
  <xs:include schemaLocation="Topology.xsd"/>
  <xs:include schemaLocation="Deployment.xsd"/>
  <xs:include schemaLocation="Control.xsd"/>
  <xs:include schemaLocation="Output.xsd"/>
  <xs:element name="experiment">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="infomation" type="Information"
/>
     <xs:element name="topology" type="Topology"/>
     <xs:element name="deployment" type="Deployment"/>
     <xs:element minOccurs="0" name="control"
type="Control"/>
  <xs:element minOccurs="0" name="output"
type="Output"/>
  </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

Fig. 3. Experiment XML Schema

Figure 3 shows the format that we have defined for an XML document describing an OpenFlow networking experiment. Such a Schema is stored in as XSD file, which refers to five external XSD files. These files contain the components of an experiment that we have described above.

```
<xs:schema version="0.1">
<xs:include schemaLocation="Node.xsd"/>
<xs:include schemaLocation="Link.xsd"/>
 <xs:complexType name="Topology">
    <xs:sequence>
      <xs:element  maxOccurs="unbounded"  minOccurs="0"
name="node" type="Node"/>
      <xs:element  maxOccurs="unbounded"  minOccurs="0"
name="link" type="Link"/>
    </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Fig. 4. Topology XML Schema

```
<xs:include schemaLocation="Interface.xsd"/>
<xs:include schemaLocation="Type.xsd"/>
 <xs:complexType name="Node">
     <xs:element   maxOccurs="unbounded"   minOccurs="0"
   name="interface" type="Interface"/>
     <xs:element name="type" type="Type" use="required"/>
     <xs:attribute       name="id"       use="required"
   type="xs:nonNegativeInteger" >
    <xs:attribute name="label"
      <xs:simpleTyep>
        <xs:restriction base="xs:string">
        <xs:pattern    value="    ([a-zA-Z](_)?)*    "/>
       </xs:restriction>
      </xs:simpleType> use="optional"/>
   <xs:attribute name="pos_x" type="xs:float" use="optional" />
   <xs:attribute name="pos_y" type="xs:float" use="optional" />
   <xs:attribute name="state" type="State" use="required" />
    <xs:simpleType name="State">
       <xs:restriction base="xs:string">
        <xs:enumeration value="CREATED"/>
        <xs:enumeration value="AVAILABLE"/>
        <xs:enumeration value="UNAVAILABLE"/>
        </xs:restriction>
       </xs:simpleType>
 </xs:complexType>
</xs:schema>
```

Fig. 5. Node XML Schema

Figure 4 shows the structure of the Topology part of a experiment description. This document also refers to external files (e.g. Node.xsd). For the sake of brevity, we just present the Node structure in Figure 5. These XML Schema samples imply that the design of OF-NEDL if a from-top-to-bottom process. The detail of each component can be redefined without affecting other component. This property provides the capacity to extend the language with minimal effort in the future.

4 Usage Scenario

To demonstrate the potential use of the OF-NEDL framework, we consider a simple but realistic networking experiment, the ping application. A minimal setup is shown in Figure 6. It consists of one switch (s1) and two endhosts (h1 and h2), each of which is configured with an IP address and connected by physical link. The logical model and XML description of the topology are shown in Figure 7. Each physical link is represented by two logical links with specified attributes. To be brief, we only show the description of s1 and ulink1.

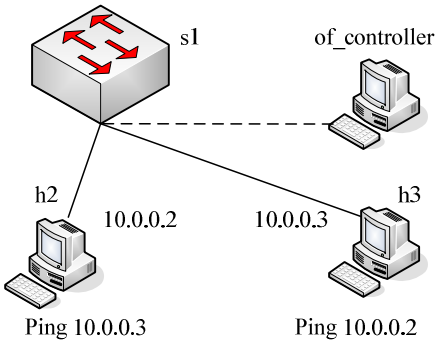

Fig. 6. The Ping Experiment Setup

The main goal of the ping application is to determine the connectivity of the network. In the experiment, the user need to implement an OF program with some basic function (ARP, STP etc.), a ping software (of course, user can use the existing software), and configure the OF switch properly (e.g. send the whole packet other than the first 128 bytes to the controller at the beginning). The deployment of the ping experiment is shown in Fig. 8. The two “config” components will add flow tables on the OF switch, so it can forward the packets correctly. By the way, the actions of manipulating the flow tables are all through the OF controller, so this configuration can be moved to the OF program.

The control of the ping experiment is simple (Fig. 9). When all the nodes are ready to run, endhosts h1 and h2 execute the commands to ping each other, and after 10 seconds, all the processes of the experiment will stop. The last part is the output component and we can imagine that the user want to show the information printed by the ping application, so the user only need to specify the “software print” as “ping” software in the output component.

```
<topology>
 <node id="s_0001" label="switch1"
   pox_x="400" pos_y="200" state="AVALIABLE">
    <interface id="s_0001_001" label="s1-eth1">
    <macAddr>fa:7a•c3:ea:c6:d1</macAddr></interface>
    <interface id="s_0001_002" label="s1-eth2">......
    </interface>
    <switch> ...... </switch>
 </node>......
   <link  id="ul_00001"  label="ulink1"     delay="0.0"
lossratio="0.0" bandwidth="100000.0">
    <from>h1-eth0</from>
    <to>s1-eth1</to>
 </link>......
</topology>
```

Fig. 7. Topology of the Ping Experiment

```
<deployment>
 <ofp>
    <name>example</name>
    <location>./example.py</location>
 </ofp>
 <ofs>
    <name>s1</name>
     <config> in_port=1,actions=output:2</config>
     <config> in_port=2,actions=output:1</config>
 </ofs>
 <software>
    <name>ping</name>
    ......
 </software>
 <ehs>
    <sw>ping</sw>
    <endhost>h2</endhost>
    <endhost>h3</endhost>
 </ehs>
</deployment>
```

Fig. 8. Deployment of the Ping Experiment

```
<control>
 <event name="AllReady">
    <command>ofnedl h2 ping c3 h3
             ofnedl h3 ping c3 h2</command></event>
    <time wait="10s"><command> ofnedl killall</command>
 </time>
</control>
```

Fig. 9. Control of the Ping Experiment

5 Conclusions and Future Work

In this paper we have investigated the general model of OpenFlow networking experiment, presented the design principle and architecture of an XML description language OF-NEDL, which aims to describe every aspects of an OpenFlow networking experiment. Since OF-NEDL is still in early design and prototyping stages, it is lack of full evaluations in the real experiment infrastructure. We will refine the design of OF-NEDL through ample evaluation in practice so it can be more practical for the researchers in the future.

References

1. Standford University: Clean Slate for the Internet, http://cleanslate.stanford.edu/
2. PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/
3. Onelab website, http://www.onelab.eu/
4. Yap, K.-K., Huang, T.-Y., Dodson, B., Lam, M., Mckeown, N.: Towards software-friendly networks. In: Proc. the First ACM Asia-Pacific Workshop on Systems, APSys 2010 (August 2010)
5. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: Enabling innovation in campus networks. SIGCOMM Computer Communication Review 38(2), 69–74 (2008)
6. NS2 network simulator, http://www.isi.edu/nsnam/ns/
7. NS3 network simulator, http://www.nsnam.org/
8. Hallagan, A.W.: The Design of XML-based Model and Experiment Description Languages for Network Simulation. Bachelor Thesis, Bucknell University (June 2011)
9. Emulab website, http://www.emulab.net/
10. Open Network Laboratory, https://onl.wustl.edu/
11. Albrecht, J.R., Braud, R., Snoeren, A.C., Vahdat, A.: Application Management and Visualization with Plush. In: Proceedings of Peer-to-Peer Computing, pp. 89–90 (2009)
12. Albrecht, J., Huang, D.Y.: Managing Distributed Applications Using Gush. In: Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S. (eds.) TridentCom 2010. LNICST, vol. 46, pp. 401–411. Springer, Heidelberg (2011)
13. GENI: Global Environment for Network Innovations, http://www.geni.net
14. ORCA website, http://www.nicl.cs.duke.edu/orca/
15. ProtoGENI website, http://www.protogeni.net
16. Rakotoarivelo, T., Ott, M., Seskar, I., Jourjon, G.: OMF: a control and management framework for networking testbeds. In: SOSP Workshop on Real Overlays and Distributed Systems (ROADS 2009), Big Sky, USA, p. 6 (October 2009)
17. The OMF Testbed Control, Measurement and Management Framework, http://www.omf.mytestbed.net
18. Lacage, M., Ferrari, M., Hansen, M., Turletti, T.: NEPI: Using Independent Simulators, Emulators, and Testbeds for Easy Experimentation. In: Workshop on Real Overlays and Distributed Systems, ROADS (2009)
19. Expedient: A Pluggable Centralized GENI Control Framework, http://www.yuba.stanford.edu/~jnaous/expedient

20. OpenFlow in Europe: Linking Infrastructure and Applications, http://www.fp7-OFELIA.eu/
21. CHANGE project, http://www.change-project.eu
22. Kanaumi, Y., Saito, S., Kawai, E.: Deployment of a Programmable Network for a Nation wide R&D Network. In: IEEE Network Operations and Management Symposium Workshops, pp. 233–238 (2010)
23. Park, M.K., Lee, J.Y., Kim, B.C., Kim, D.Y.: Implementation of a Future Internet Testbed on KOREN based on NetFPGA/OpenFlow Switches. In: NetFPGA Developers Workshop, Stanford, CA, August 13-14 (2009)
24. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: NOX: Towards an Operating System for Networks. SIGCOMM Comput. Commun. Rev. 38, 105–110 (2008)
25. Beacon, http://www.openflowhub.org/display/Beacon/
26. SNAC OpenFlow Controller, http://snacsource.org/