An OpenFlow-based Testbed for Information Centric Networking

N. Blefari-Melazzi[1], A. Detti[1], G. Mazza[1], G. Morabito[2], S. Salsano[1], L. Veltri[3]
[1]*University of Rome Tor Vergata / CNIT, Rome, Italy*
[2]*University of Catania / CNIT, Catania, Italy*
[3]*University of Parma / CNIT, Parma, Italy*

Abstract: Information-centric networking (ICN) is a novel networking paradigm which is attracting increasing attention by both academic and industrial researchers. In fact, it promises to provide technological solutions that best fit with the way in which Internet is actually utilized. Assessment of proposed solutions require appropriate experimental testbeds. In this context OpenFlow, which has been developed to enable the deployment of novel networking solutions in the actual network infrastructure, represents a valuable tool. Accordingly, we are currently implementing an ICN solution – called *CONET* – for OpenFlow networks. The solution will be deployed in two testbeds, part of larger experimental OpenFlow facility distributed across Europe realized by the EU funded OFELIA project. In particular one testbed will be based on the Open vSwitch platform while the other will be deployed on NetFPGA platforms. Our implementation has been designed to easily support other ICN solutions with simple modification of the code. Basic ICN functionality that are specifically addressed in our implementation are data naming, route-by-name, and in-network caching.

Keywords: Information-centric networking, software defined network, OpenFlow.

1. Introduction

In recent years both the industrial and academic communities have focused increasing attention on the so called *information-centric networking* (ICN) paradigm, which is expected to be a fundamental component of the Future Internet [1], [4].

Motivations of the ICN paradigm lay in the evident fact that [1] "*people value the Internet for* what *content it contains, but communication is still in terms of* where*"*. In other words the Internet was designed to support exchange of data between hosts, whereas most of Internet usage relates to obtaining a desired content regardless of the specific hosts that store it. Such mismatch between the principal use of the Internet and the current protocols and architectures has several disadvantages in terms of persistence, availability, and authenticity of data, which become more and more critical as user mobility and importance of data authenticity increase [2].

The large interest in the ICN paradigm, mentioned above, is demonstrated by the large number of ongoing research projects focusing on ICN worldwide, by the increasing number of workshops dedicated to ICN, and by the creation of an "*Information-Centric Networking Research Group*" (ICNRG, [5]) within the Internet Research Task Force (IRTF). Immediate consequence of such interest is a large volume of solutions for ICN proposed in the last few years [2], [1], [6]. Comparison of proposed solutions, which often are based on approaches significantly different one from the other, is very difficult because metrics and experimental settings should be uniquely defined in order to achieve fair comparisons.

This work was supported in part by the EU under the FP7 projects "OFELIA" and "CONVERGENCE"

While this is among the objectives of ICNRG [5], it must be mentioned that remarkable results could be achieved by deploying the experimental solutions into real network infrastructures. Unfortunately, this is not done usually because it is extremely complex (and even dangerous) to deploy experimental solution at the hearth of critical infrastructures such as the Internet.

Generally speaking, such criticality has been a major barrier to the introduction of novel solutions in the Internet in the last few decades, so that network architecture and protocols currently utilized by the Internet have been defined several decades ago. To overcome this problem, OpenFlow has been recently proposed [3]. OpenFlow enables researchers to deploy novel switching/networking solutions in the actual network infrastructure while preventing problems to be created to actual data traffic traversing the network. In fact, OpenFlow enables researchers to implement novel architectures and protocols via software and to deploy them in actual network switches/routers. Network resources are *sliced* and operations running in different slices do not interfere each other which guarantees a sort of isolation between slices and therefore, prevents novel solution to create problems to normal traffic.

Accordingly, OpenFlow can be regarded as the first significant instantiation of the so called *Software Defined Network* concept which envisions a physical separation between datapath (usually, realized in hardware) and network control (usually, realized via software). As a result of such separation new networking solutions can be tested, and cost of infrastructure can be reduced significantly. New products released by several network equipment manufacturers support OpenFlow and large OpenFlow experimental testbeds have been deployed worldwide – see the testbed at Stanford University (http://cleanslate.stanford.edu) or the OFELIA facility (http://www.fp7-ofelia.eu), for example.

In this paper we discuss an ICN solution for OpenFlow based networks and describe the experimental testbeds we are deploying. ICN characteristics which will be specifically addressed regard data naming, route-by-name, and in-network caching. In Section 2 we provide a background on ICN and on OpenFlow. Section 3 introduces the CONET solution that we have chosen as basis for the integration with OpenFlow. Section 4 specifically describes how the CONET ICN solution can be implemented on top of OpenFlow. Finally in Section 5 some concluding remarks are drawn.

2. Background

2.1 Information-Centric Networking

In the recent past several solutions have been proposed for Information-Centric Networking. As for their overall approach, the “clean slate” approaches aims at fully replacing the existing IP layer, i.e. assuming that the ICN layer will sit over layer 2 networks like Ethernet. On the other hand, the “overlay” approaches considers to run ICN over the existing IP based networks. This means that ICN information units are tunnelled within TCP or UDP flows running over IP. Recently, an “evolutionary” approach [8] has been proposed which foresees to extend IPv4 and IPv6 to support ICN using a new option to be carried in IP packets headers.

While the ICN approaches can be extremely different each other, key components for all of them are: naming strategy, route-by-name policy, and in-network caching solutions. In ICN systems, data items – which can have several instantiations in the network – are uniquely identified by a *name*. Such name should be reported in messages relevant to the retrieval and transportation of the data item. As a consequence the policy utilized to select and represent such name are extremely important. In fact, mnemonic, variable length names

provide higher flexibility and are more human-friendly. However, they also require variable field size in packet header (which results in slower packet handling within the network) and complex mechanisms to ensure name uniqueness. The alternative approach envisions, instead, flat, constant length, computer generated names. Such names cannot be memorized by humans easily and therefore, require a network service which is responsible to map a human friendly, descriptive name in the actual name[1].

Route-by-name indicates the mechanisms used to retrieve a content, when a user has made a content request providing the name of the requested content. These mechanisms can be split in "forward-by-name" and "content routing". Forward-by-name refers to the operation of relaying an incoming content request to an output interface, based on the name of target content. Instead, content routing indicates the operation of disseminating information about location of contents. In this context, two different approaches can be clearly distinguished depending on whether the routing logic is distributed in each node in the communication path or executed in appropriate network elements that will communicate their decisions to the relevant network nodes through appropriate signalling.

In-network caching is another key component of all ICN solutions, even if the debate about its effectiveness in real networking settings is still on-going [7]. Here again several design options can be taken. For example, storage can be distributed in some (or all) of the network nodes or can be centralized in one (or several) specific server farms that are utilized by a portion of the network. Furthermore, a large number of caching policies are available and the most appropriate should be identified.

The objective of this paper is to propose an implementation of ICN in OpenFlow that can potentially accommodate a large set of the solutions proposed so far for the above functionality.

2.2 OpenFlow

Software Defined Networking (SDN), which envisions a separation between the network components that are responsible for packet forwarding from the components that are responsible for the network control, is the most promising environment for implementing and testing open and flexible ICN solutions in real networks. Accordingly, we have focused our attention on OpenFlow which provides a set of standardized, open interfaces for the definition and management of a Software Defined Network.

In OpenFlow the network elements responsible for packet forwarding are denoted as OpenFlow Switches (OF Switches), whereas the network elements responsible for network control are called Controllers. OF Switches and Controller(s) interact by exploiting the OpenFlow Protocol which defines a set of primitives that are exchanged through an SSH connection.

Upon arrival of a data packet OF Switches check whether there is a matching between certain fields contained in the header[2] of the data packet and the entries that are stored in its *flow tables*. If this is the case, then the appropriate *action* (reported in the flow table) is executed, otherwise the packet header is sent to the Controller and the packet is held until the Controller notifies what is the action to be executed on such a packet. Possible actions are: the forwarding through one or more switch network interfaces, the modification or dropping of the packet, and several others that have been defined [9]. When the Controller receives the header of the packet it decides the action to be executed on such packet based on some policies that can be programmed via software by the network managers. Accordingly, all intelligence is concentrated in the Controller that should have an updated view of what is the current status of the network elements it is responsible for.

1 Such servers have the same goals of the DNS, in the current Internet.

2 OF Switches consider the headers at the second, third, and fourth level of the protocol stack.

Communication and computing resources of network elements can be divided in several *slices* and different policies can be defined by the controller for each slice. This enables the coexistence of several networking policies and architectures in the same infrastructure.

Several implementations of OpenFlow switches are available both in software [10] and hardware [11]. Furthermore, several open source implementations of the controller are available as well (for example, see the NOX implementation [12]). Public experimentations of OpenFlow are ongoing, the EU funded OFELIA project has recently deployed a large OpenFlow based testbed across several "islands" in Europe [13], open for experiments. Our goal is to design and implement an ICN solution on top of the OFELIA OpenFlow testbed.

3. The CONET solution for Information Centric Networking

Among the different ICN proposals, we have selected CONET [6] and worked on its implementation on OpenFlow infrastructure. This section provides an overview of the CONET network architecture and protocols.

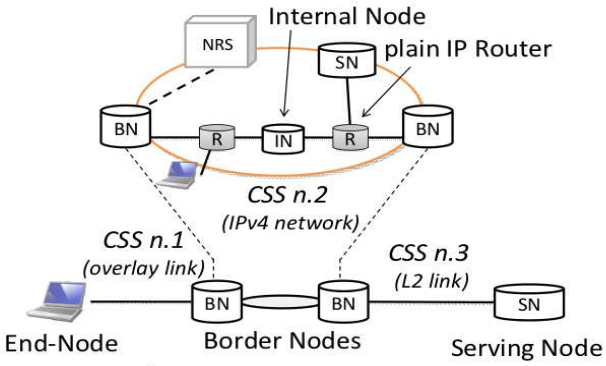


Figure 1: CONET network architecture.

As shown in Figure 1, a CONET network consists of a set of end nodes and serving nodes interconnected by CONET Sub Systems (CSS). A CSS is defined as a generic network with homogeneous networking technology and homogeneous native addressing space. In our case one of the CSS will be an OpenFlow network.

CONET nodes exchange CONET Information Units (CIUs), which are used to convey both requests of named-resources, called *interest* CIUs, and chunks of named-resources themselves (i.e., part of files, videos, etc.), called *named-data* CIUs. A CONET SubNet interconnects two or more CONET nodes, providing transfer of CIUs by using an under-CONET technology, such as point-to-point Layer 2 links, Layer 2 networks or overlay links (e.g. UDP over IP). To best fit the transfer units of an under-CONET technology, both interest-CIUs and named-data CIUs are carried in small CONET data units named *carrier-packets*.

In a CSS we can find Border Nodes and Internal Nodes. *Border-Nodes* interconnect different CSSs and forward carrier-packets by using CONET routing mechanisms, reassemble carrier-packets, cache the related named-data CIU, and may send back cached named-data CIUs. Optionally, *Internal Nodes* can be deployed inside a CSS to provide additional in-network caching facilities. A *Name Routing System* (NRS) Node is needed in order to assist CONET in performing routing operations.

Hereafter we describe the operations executed in order to deliver a chunk of named data to the end node that has requested it. Operations begin with the end node that generates an interest CIU which includes the network-identifier – i.e., the name of the resource – named NID; the interest CIU is encapsulated in a carrier-packet, which we denote as iCPx (Interest Carrier Packet x). Name-based forwarding engines in the End-Node and intermediate Border-Nodes *forward-by-name* the packet iCPx upward the proper Serving-Node.

Forward-by-name means that, on the base of the network-identifier contained in the carrier-packet iCPx, a name-based routing-engine singles out the CSS address[3] of the next upward Border-Node toward the Serving-Node. Then, the name-based routing engine encapsulates the carrier-packet iCPx in a data-unit of the underlying CSS technology and uses the CSS address as the destination address. The internal-nodes (if any) receiving the Carrier Packet iCPx forward it by using the underlying routing (e.g. IP RIB). Nevertheless, before forwarding the packet, they parse it and send back the requested named-data CIU if this is available in their cache. Note that this is executed by border nodes as well. When a node storing the content is reached (it can be the origin Serving Node or a caching Border Node or Internal Node), the named-data CIU is encapsulated in a carrier-packet, here denoted ndCPx (named data Carrier Packet x). The carrier-packet ndCPx follows the same path of the carrier-packet iCPx, but in the downward direction and will reach the requesting end-node. The Interest CIUs can designate the reverse path for the named-data CIU using two mechanisms. The first mechanism is the same used in in CCN [1]: the Border Nodes store the "pending" requests in a table called "Pending Interest Table", recording for each Interest CIU the previous hop. The second mechanism is based on a field named path-state that can be inserted in the Interest CIUs. The path state can be used by a border node to record the CSS address of previous hop in the "*upward*" path toward the serving-node. The recorded path state will be copied in the data CIU so that a source routing is used for the downward path. Note that in general a combination of information stored in the Pending Interest Table and in the path state can be used. Finally we note that when the named-data CIUs are sent, all the border-nodes and internal-nodes in the downward path may cache them, according to their policies and available resources.

CONET functionality can be integrated in IP networks by using the *IP options*, which have been defined for both the IPv4 and IPv6 in [8]. The CONET IP option carries the *name* of the data and allows IP routers to be extended with ICN functionality in a backward compatible way.

4. CONET implementation in OpenFlow networks

In this section we describe the solution we have designed to experiment CONET in OpenFlow networks and the preliminary development and testbed results. We based our work on the assumption to use the OpenFlow 1.0 specification, which is available in several equipment and in particular in the testbed offered by the OFELIA project. Note that OpenFlow 1.2 specification has just been announced by Open Networking Forum, however it has not been officially released, and no equipment is currently available in the OFELIA testbed. We will take into account OpenFlow 1.2 in future researches. The assumption to consider OpenFlow 1.0 rules out the possibility to inspect the packet header at arbitrary positions, for example taking into account the newly defined IP options. On the contrary, only a predefined set of fields in the packet headers can be inspected. The solution we envisaged is to map the content name carried in the IP option (denoted as ICN-ID) into a tag transported in a field that can be inspected in OpenFlow 1.0 equipment. We choose the "transport level" ports field that is the 4 bytes that in TCP and UDP identify the source and destination port. Note that the CONET Carrier Packets are transported with a newly defined IP transport protocol, different from UDP and TCP. This approach foresees to have gateway nodes between a non-OpenFlow CONET network and OpenFlow based CONET network, that opportunely adapts ICN CPs. For simplicity we can assume that an OpenFlow based CONET network corresponds to a CONET CSS as shown in Figure 1. A "gateway" Border Node that receives an Interest Carrier Packet will implement the mapping between

3 A "CSS address" is an interface address consistent with the CSS underlying technology (for example, it is an IP address if the CSS is an IP network.

the ICN-ID (which could be of fixed or variable length depending on the ICN naming schema) into a 4 byte tag to be carried in the "transport level" ports field. This mapping does not need to be reverted in the outgoing gateway border node, because the full name is still carried in the ICN-ID field in the IP option, so the outgoing border node only need to remove the tag. The mapping between ICN-IDs (the content names) and the 4 bytes tag must be identical in all border nodes, therefore we assume that a NRS node can perform it. The mapping will be dynamic, therefore the tags can be reused and will expire if not used. Therefore, the number of 2^32 different tags (as the tag is 4 bytes long) does not define the maximum number of different content names available, but the number of "active" contents that are being requested and distributed in a CONET Subsystem, which can be reasonable. The mapping between ICN-IDs and tags will be cached in the border nodes, therefore only the first request for a content will be subject to the mapping request to the NRS. Note also that in the proposed CONET solution the border nodes will request to the NRS the next hop when receiving the first request for a content. The same message used for route-by-name lookup can be used to request the ICN-ID to tag mapping.

This approach can also be applied to support other ICN solutions in an OpenFlow based network section. Considering for example the CCN [1] solution, it would require to introduce a CCN gateway node that is able to map the CCN content name into the fixed size tag (possibly with the help of a server/controller) and then to carry the CCN interests and data packets in the newly defined IP transport protocol.

4.1 Protocol operations

In this section we report the operations executed when an Interest Carrier Packet (CP) or a Named-data CP traverse an OpenFlow-based CSS. Let us suppose that a Border Node (BN_{In}) of an OpenFlow CSS receives a packet. If it is not a CONET packet (the IP protocol is not "CONET"), then the packet is managed by the OpenFlow network using OpenFlow standard mechanisms. In case it is a CONET packet (the IP protocol is set to "CONET"), two cases are possible: i) it is an Interest CP or ii) it is a Named-data CP.

If it is an Interest CP, the Border Node BN_{In} has to identify the Border Node towards which the Interest CP should be forwarded (let us denote such Border Node as BN_{Out}) and has to give the Interest CP an appropriate format so that it can be processed by OpenFlow switches rapidly. In order to achieve the first goal, BN_{In} queries the NRS which runs in the node executing the Controller, that we call node C.

To this purpose the header of the packet (containing the IP options) is sent to the node C. The NRS running in C determines whether the same interest has been issued by some other end node recently. If this is not the case then, the NRS assigns a 32 bit flow identifier *f* to such interest and identifies the most appropriate output border BN_{Out}. Otherwise, the node C assigns the flow identifier which has been already assigned to the interest as well as the same output border node, BN_{Out}. Then a rule is set with flow identifier equal to *f* towards BN_{Out}, and an entry in the flow table of node BN_{In} is created that forwards the Interest CP through the network interface towards BN_{Out} or towards the Cache Server if the requested content is stored in the local cache.

The input Border Node BN_{In} uses the information received by the NRS to encapsulate the Interest CP in a packet with a format that can be processed by OpenFlow Switches rapidly. BN_{In} encapsulates the Interest CP in an IP packet in which the IP protocol is set to "CONET", the "transport layer ports" are set equal to the flow identifier *f* and the destination IP address is set equal to the IP address of BN_{Out}.

Then such new packet is forwarded to the OF Switch hosted in BN_{In}. The OF Switch will forward the Interest CP packet along the appropriate network interface, or towards the local cache server. More specifically, if the content is stored in the cache, the content is

transmitted towards the requesting end node and the Interest CP is no longer propagated. All OF Switches in the path towards the output Border Node BN_{Out} will handle the same packet in analogous way. The Border Node will restore the original Interest CP from the received packet by removing the tag and will forward it in the following CSS towards the Serving Node.

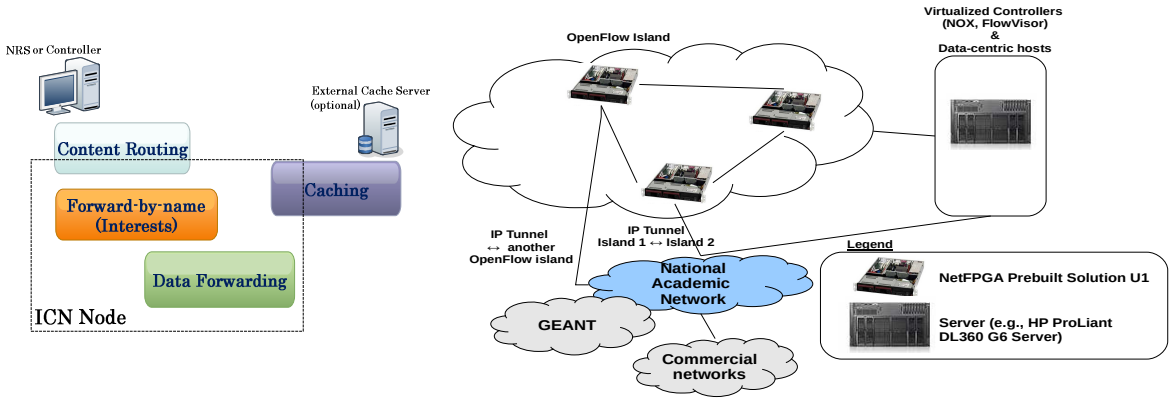
If the packet entering the OpenFlow-based CSS is a Named-data CP either it contains information about the address of the most appropriate output Border Node (BN_{Out}) or a "Pending Interest Table" in the node will provide this information, as discussed in Section 3. Accordingly, the Border Node BN_{In} encapsulates the named data CP in an IP packet and sets the IP destination address equal to the address of the output Boder Node, BN_{Out}.

Internal Nodes in the path between the input and output Border Nodes may decide to store copy of such data in their local cache. To this purpose appropriate caching strategies are possible. We assume that content of the cache is updated according to the Least Recently User (LRU) algorithm. Also in this case, the output Border Node will restore the original named data CP from the packet and forward it towards the requesting end node.

4.2 Software architecture

As shown in Figure 2(a) the architectural components of our prototype are:

- **Forward-by-name:** it is applied to Interest packets, it is the mechanism used by ICN nodes to relay an incoming content request to an output interface. The output interface is chosen by looking up a "name-based" forwarding table.
- **Data Forwarding**: it is the mechanism that allows the content to be sent back to the device that issued a content request. Data forwarding cannot use the Forward-by-name mechanisms, because the devices are not addressed by the content routing plane of an ICN. Therefore, an ICN requires two different forwarding strategies to forward content requests and to deliver the data.
- **Content Routing:** it is the mechanism used to disseminate information about location of contents, so as to properly setup the name-based forwarding tables. For instance, content routing could re-use IP routing mechanisms, where name prefixes are distributed instead of IP prefixes. Content routing is one of the assets of ICN, as a provider could use content routing to improve the efficiency and reliability of content access in its network.
- **Caching:** it concerns the ability of ICN nodes to cache data and to directly reply to incoming content requests rather than forwarding them towards a serving node.



(a) (b)

Figure 2: Implementation architecture (a) and testbed configuration (b).

4.3 Experimental testbed

In Figure 2(b) we show the testbed configuration. Each of the two testbeds consists of three fully meshed OpenFlow Switches (in one island we have NetFPGA OpenFlow switches in the other we have Open vSwitch). Such switches are connected to a server where the OpenFlow NOX Controller is executed. In order to achieve higher flexibility we run the NOX Controller (and the FlowVisor [9]) in a virtual environment. The islands will be fully operational OFELIA islands, connected to the OFELIA experimental testbed [13].

5. Conclusions

Assessment and comparison of ICN solutions require appropriate experimental testbeds. Accordingly, we are developing a framework that consists of a prototype of an ICN solution called CONET and an experimental platform of OpenFlow switches. The experimental platform includes two different OpenFlow *islands* connected with each other through our national academic network. The two islands will be connected to other OpenFlow islands distributed across Europe by means of the GEANT network. Our ICN prototype has been designed in such a way that it can be easily modified to assess other solutions proposed for ICN systems.

On-going work includes the identification of the most appropriate caching strategy, the definition of a methodology which enables the serving node to signal the priority that should be considered when taking decisions about caching of a given content, and the design of scalable solutions for the Name Routing System. An additional interesting thread of research is related to the evolution of OpenFlow API to explicitly support ICN. When new OpenFlow releases will support full packet inspection, we will be able to map the ICN based operations of a future ICN enabled switch into new OpenFlow notifications and commands exchanged between the OpenFlow switch and controller.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M.F. Plass, N.H. Briggs, and R. L. Braynard, “Networking Named Content”, Fifth ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2009.

[2] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. “A Data-Oriented (and Beyond) Network Architecture”. *Proc. of ACM Sigcomm.* August 2007.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”. *White paper*. March 2008 (available at: http://www.openflow.org).

[4] D. Kutscher, H. Flinck, and H. Karl. “Information-Centric Networking: A Position Paper”. *Proc. Of 6th GI/ITG KuVS Workshop on Future Internet.* November 2010.

[5] http://wiki.tools.ietf.org/group/irtf/trac/wiki/icnrg

[6] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini. “CONET: A Content-Centric Inter-Networking Architecture”, *Proc. of ACM Sigcomm – ICN 2011.* August 2011

[7] D. Perino and M. Varvello. “A Reality Check for Content-Centric Networking”, *Proc. of ACM Sigcomm – ICN 2011.* August 2011.

[8] A. Detti, S. Salsano, N. Blefari-Melazzi. IPv4 and IPv6 Options to Support Information Centric Networking. *Internet draft* (draf-detti-conet-ip-option-02). November 2011.

[9] B. Pfaff, *et al.* OpenFlow Specification – *Version 1.1.0*. February 2011

[10] http://www.openflow.org/

[11] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown. “Implementing an OpenFlow switch on the NetFPGA platform”, *Proc. of ACM/IEEE ANCS '08*. November 2008.

[12] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: Towards an Operating System for Networks”, *ACM Computer Communication Review. pp.* 105-110. July 2008

[13] A. Köpsel and H. Woesner, “OFELIA – Pan-European Test Facility for OpenFlow Experimentation”, *Lecture Notes in Computer Science.* Vol. 6994/2011. 2011