IMPLEMENTING *NDN* USING *SDN*: A REVIEW OF METHODS AND APPLICATIONS

SHIVA ROWSHANRAD*, MOHAMAD REZA PARSAEI AND MANIJEH KESHTGARI

Information Technology Department,
Shiraz University of Technology, Shiraz, Iran.

**corresponding author: shiva.rrad@gmail.com*

(Received: 11th Oct. 2015; Accepted: 20th Jan. 2016; Published on-line: 30th Nov. 2016)

ABSTRACT: In recent years, many claims have been made about the limitations of today's network architecture, its lack of flexibility and its inability to respond to ongoing changes and increasing users demands. In this regard, new network architectures are proposed. Software Defined Networking (SDN) is one new architecture that centralizes the control of network by separating control plane from data plane. This separation leads to intelligence, flexibility and easier control in computer networks. One of the advantages of this framework is the ability to implement and test new protocols and architectures in actual networks without any concern of interruption. Named Data Networking (NDN) is another paradigm for future network architecture. With NDN the network becomes aware of the content, rather than just transferring it among end-points. NDN attracts researchers' attention and is known as the potential future of networking and internet. Providing NDN functionalities over SDN is an important requirement to enable the innovation and optimization of network resources. In this paper, a description of SDN and NDN are given, and methods for implementing NDN using SDN are introduced. Various methods are compared from hardware methods requiring changes in switches, to software methods such as using proxies, to methods that take advantage of unusable IP header fields as NDN headers. The advantages and applications of implementing NDN over SDN for routing, virtualization, traffic management, data centers, and heterogeneous networks are also pointed out.

ABSTRAK: Sejak beberapa tahun kebelakangan ini telah timbul banyak dakwaan mengenai batasan senibina rangkaian komputer hari ini, kekurangan fleksibiliti dan keupayaan untuk bertindak balas kepada perubahan yang berterusan dan permintaan pengguna yang semakin meningkat. Dalam hal ini, seni bina rangkaian baru adalah dicadangkan. Software Defined Networking (SDN) adalah salah satu daripada seni bina baru yang memusatkan kawalan ke atas rangkaian dengan mengasingkan satah kawalan dari satah data. Pengasingan ini membawa kepada kecerdasan, fleksibiliti dan kawalan lebih mudah dalam rangkaian komputer. Salah satu kelebihan rangka ini adalah keupayaan untuk melaksanakan dan menguji protokol dan seni bina baru dalam rangkaian sebenar tanpa sebarang gangguan. Named Data Networking (NDN) adalah satu lagi paradigma untuk seni bina rangkaian masa depan. Dengan NDN rangkaian mengetahui kandungan, bukan hanya memindahkannya antara mata akhir. NDN menarik perhatian penyelidik dan dikenali sebagai potensi masa depan bagi rangkaian dan internet. Menyediakan fungsi NDN lebih dari ISDN adalah satu keperluan yang penting bagi membolehkan inovasi dan pengoptimuman sumber rangkaian. Dalam kertas ini mulanya kami menerangkan tentang SDN dan NDN, kemudian kami memperkenalkan kaedah untuk melaksanakan NDN menggunakan SDN. Kami membandingkan methodes yang memerlukan perubahan perkakasan dalam suis, methodes perisian yang menggunakan proksi dan methodes menggunakan bidang pengepala IP yang tidak lagi boleh digunakan sebagai pengepala NDN. Kami juga menunjukkan aplikasi dan

kelebihan mengguna NDN berbanding dengan SDN untuk routing, virtualisasi, pengurusan trafik, pusat data dan rangkaian heterogen.

KEYWORDS: *software defined networking; named data networking; future internet; openflow*

1. INTRODUCTION

Today's network architecture was proposed between the 1960's and 1970's [1]. So far, there have been many changes in internet usage which require it to migrate to new network architectures. Changes include new services and technologies such as cloud computing and big data, the diversity of IT users, especially mobile users, etc. Furthermore, current users tend care for their content and information rather than where and how to access such content [2, 3].

In recent years, new architectures and frameworks have been proposed and investigated to solve current network issues and limitations. As an example, Software Defined Networking (SDN) is proposed to create highly programmable computer networks. In SDN, the separation of the control plane from data plane gives network operators the ability to introduce new protocols and services to the network without any changes in the network hardware or infrastructure [4]. One of the most popular ways to deploy SDN is by implementing it based on an open standard protocol, named OpenFlow. In SDN/OpenFlow, network devices are simple forwarders named OpenFlow switches, while a centralized controller is responsible for making routing and control decisions [4, 5].

Named Data Networking (NDN) is another paradigm for new network architecture which has recently attracted many researchers' and academia's attention. This architecture is proposed to solve current network issues such as high bandwidth cost due to point-to-point traffic and content delivery networks (CDNs). There is a difference between today's network principles and its applications. To deal with this difference, NDN introduces an architecture that focuses on content access and delivery instead of host-centric communication. NDN introduces services such as in-network caching and service classification based on content [6]. The functionalities and advantages of NDN can be truly realized when the network's infrastructure is dedicated to this architecture. Furthermore, to improve network operation and efficiency, there is a need for new solutions for deploying new architectures incrementally while they are compatible with current IP networks. Therefore, developing NDN devices in current networks is an important issue.

SDN brings the opportunity for innovations and development of new network architectures. This can be the mainspring of implementing NDN functionalities using SDN. SDN also facilitates the implementation of NDN without the need for new hardware, hence decreasing infrastructure expenses.

In this paper, the methods and applications of implementing NDN using SDN are discussed. First, an overview of SDN and NDN are given in section 2 and section 3 respectively. Then, in section 4 existing methods for implementing NDN over SDN are explained and compared. In section 5, the application of NDN over SDN is introduced. Finally, section 6 concludes the paper.

2. SOFTWARE DEFINED NETWORKING

In SDN, the control plane and data plane are separated, which makes the networks control more flexible. In current networks, routers include both control plane and data plane. The control plane is responsible for routing computation and decisions while the data plane is responsible for forwarding packets. In SDN, the control plane is centralized into a server called the controller. The controllers are modular network operating systems. New services such as monitoring and new routing protocols can be added to network by adding different software modules to the controller, hence analyzing and controlling network behaviour. For this purpose, the controller also needs to communicate with underlying switches using standard protocols such as OpenFlow [4, 7].

Figure 1 shows the SDN architecture which has three layers: an infrastructure layer consisting of OpenFlow switches, a control layer in which the controller is placed, and an application layer in which business applications can be placed. Aside from these three layers, there are two interfaces: the first is a southbound interface such as OpenFlow which standardizes the communication between the infrastructure and control layers; the second is a northbound interface which is used for communication between the controller and application layers [2, 4].

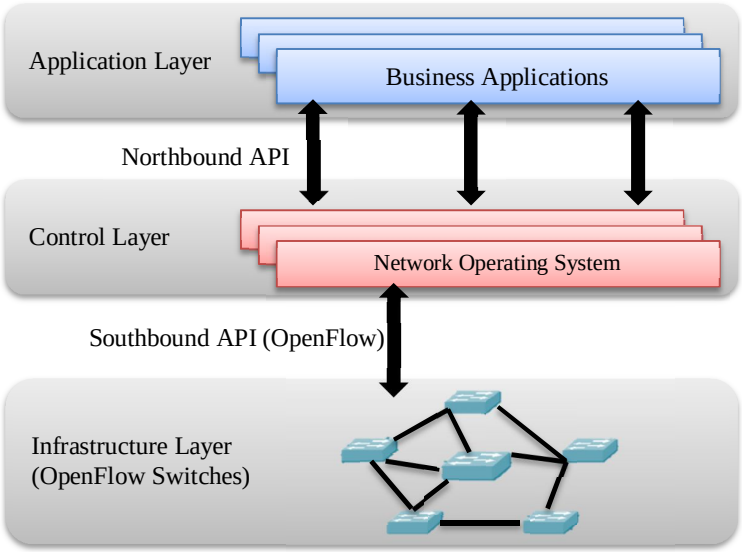

Fig. 1: SDN architecture.

In the infrastructure layer, OpenFlow switches are simple forwarders which communicate with the controller via OpenFlow protocols. This communication is achieved through a secure channel such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS). The switches have one or more flow tables. Each flow table consists of flow entries for forwarding flows. Each entry includes three main parts [7, 8]:

- Matching field: The information of layer 2 to layer 4 header of packet, ingress port and metadata. This field is for matching the entries to arriving packet.
- Action: Set of operations and instructions to be done on the packet after matching process.

- Counters: Statistical information about a flow such as received and sent bytes/packets, dropped packets, or duration.

The entries can be added, deleted, or updated by the controller through OpenFlow messages. Upon receiving a packet, the switch looks into the flow table to find a match based on the matching field. If a match is found, the specified action will be done on the packet and counters will be updated. If a table-miss happened, the switch will encapsulate the received packet as a packet-in message and send it to the controller for decision making. Then the controller informs the switch about its decision using a packet-out message [4, 5].

3. NAMED DATA NETWORKING

As mentioned before, in NDN, the concentration is on content delivery instead of physical location of content. This architecture is based on named content and does not recognize any host entity. Also, in this framework, the contents have unique names similar to a URL, instead of IP addresses. The names consist of human readable parts with different lengths [6, 8]. NDN has the functionality of in-network caching. Upon receiving a data packet the router can cache it into its content store. The communication starts from the consumer via an interest packet which contains the request for desired content and its name. When an interest packet gets into a router, the router looks up the content store based on longest prefix matching (LPM) for the desired content. If the content is available, the router sends it to the consumer from the same interface that received the interest and discards the interest packet. If no content matches the interest packet, the router looks into another table called the Pending Interest Table (PIT) which contains the list of interests waiting for data. If the look up process was successful, the ingress interface of the interest packet will be added to the related table entity; otherwise the router will look in the forwarding information base (FIB) table to forward the interest packet to the producer. If no related entity is found in FIB, it means that the router neither has the content, nor knows how to get to it; so it discards the interest packet. The data packet follows the interest packet chain of entities in PITs to get to the consumer [3].

4. IMPLEMENTING *NDN* USING *SDN*/OPENFLOW

As mentioned before, NDN is a new network architecture proposed for solving limitations and issues in current networks. NDN has functionalities and advantages such as in-network caching, data-centric security (by signatures on all packets), multipath forwarding, self-regulation (by flow balancing between interest and data packets) [6, 9]. These functionalities have the greatest effect in the network when NDN infrastructure and devices are available in the network.

One of the opportunities from adopting SDN is the ability to deploy and test new network architectures. Implementing NDN using SDN makes it possible to define NDN's functionality in the network without the need for new hardware, thereby decreasing infrastructure expenses. Overcoming the weaknesses of NDN or optimizing its capabilities and resources, such as in-network storage, is another motivation for deploying NDN over SDN. Many projects proposed different methods for implementing NDN over SDN. Four groups of these methods will be investigated in this paper [1, 8, 10-15]:

- Changing OpenFlow switches

- Use of Hash functions to hash packet names into IP header fields
- Using proxy to communicate between OF switch and CCNx (an NDN implementation reference, CCN stands for Content Centric Network which is the earlier name of Named Data Network)
- Using the IP Option field of a packet as the name field

In [1] an architecture named ContentFlow, is proposed which is shown in Fig. 2. In this architecture the OpenFlow switches must change to be able to process NDN packets and support its functionalities.

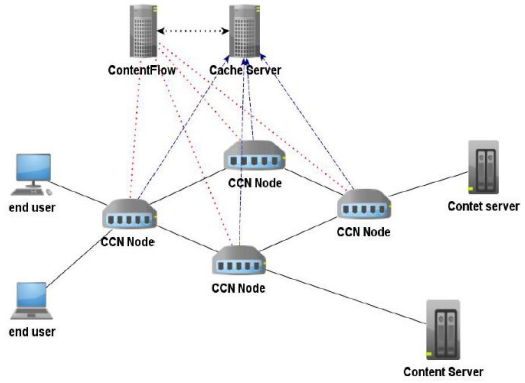

Fig. 2: ContentFlow Architecture Model [1].

The CCN node in Fig. 2 is the modified OpenFlow switch which is programmable through the network controller and supports NDN transactions. This node includes FIB, PIT, and content storage. The ContentFlow node is the controller that communicates with CCN nodes and the content server to manage the network. The CCN nodes transfer the copy of the contents they have, and their updated information, to the content server periodically. The controller can query the content list and their information from the content server. This gives the controller the ability to take a global view of the network. The content server is also responsible for making decisions about the location of content in the network and their caching duration in the CCN node [1]. The strong point of this architecture is that while it can deploy all functionalities of NDN, it also supports the advantages of SDN such as a centralized global view of the network from the controller. Furthermore, it creates the ability to add new protocols and services to the network, independent of changing hardware devices, which is a strong advantage since NDN is still evolving and may face many changes. However, changing OpenFlow switches to build CCN nodes is dependent on hardware changes that are time consuming and expensive. It is therefore better to use existing available elements as much as possible in NDN architecture testing phases. The problem is that SDN forwarding and routing is based on flows and their layer 2-4 header information, while NDN's is based on content names.

In [16] a method is proposed that is based on hashing content names in the IP packet's address field. To implement this method, User Datagram Protocol (UDP) and OpenFlow 1.0 are used. The NDN packets are encapsulated into the payload part of UDP packet. Two different port numbers are dedicated to interest and data packets. This makes the nodes able to distinguish between these two NDN packets and also recognize IP packets from NDN packets; hence the switches can work with both NDN and IP

architectures. The NDN routers should be able to perform LPM on packet names, but there is no rule in OpenFlow protocol for matching packet payloads. As a result, the names are hashed and placed into the IP field in the packets header. Since each part of a name would be hashed into a 4 bit number, a name can consist of 8 parts at most. One problem occurs because the names in NDN can have different lengths, and studies show that 99.9 percent of names have fewer than 30 parts; so in most cases, 8 parts are insufficient. The collision probability is also dependent on the number of bits. The greater the number of bits for each part of the name, the lower the collision probability. This means that there is a trade-off between the number of the parts and the number of bits for each part. A better solution is to consider OpenFlow 1.3 for exploiting IPv6 or MAC fields.

This paper [16] uses software caching in the controller as in-network caching that clearly has a better performance than hardware caching in routers. Instead of PIT, FIB, and content store, it considers a state flag in the controller for each packet [16]. Although this method makes the controller aware of all content in the network, it doesn't have the most important advantage of NDN architecture named "on-path caching". On-path caching brings the contents closer to consumers, and lowers the bandwidth consumption and latency. However, on-path caching also has shortcomings. Due to high content replication in nodes, the maximum amount of content that can be cached inside the domain is limited. To solve this issue in [11], a new strategy is proposed for "off-path caching" in NDN using SDN functionalities. This strategy lowers the content replication and increases the content hit-rate. For integrating SDN and NDN, this paper proposed a method in which the network switches are composed of an OpenFlow switch and CCNx that communicate with each other via a proxy. The proxy maps the CCNx and OpenFlow switches and hashes the NDN content names into fields that can be processed by OpenFlow switches. The advantage of this method is that there is no need to change CCNx or OpenFlow switches and the proxy can be updated easily. The OpenFlow hardware cache can be used as FIB while the CCNx daemon can be used as PIT and content store. But the proxy can decrease the network performance due to extra operations it should do. For solving this issue, three modules where added to the controller. The first module is a measurement module that periodically identifies the most popular contents by querying the switches. The second module is an optimization module that finds the best location for caching content based on latency and congestion. The third module is a deflection module that builds a mapping between the hashed name of the content and the outgoing interfaces where it was cached. The evaluation results show that the proxy decreases the efficiency by only 5% in the worst case [11, 12].

In CoNet architecture, which is implemented over the OFELIA test-bed, the IP option is used as the name field. Due to the inability of OpenFlow in processing IP options, the name is also hashed and tagged into the first 4 bytes of the UDP header. As the CoNet packet is placed into the UDP packet, the IP protocol is 17 for both the UDP and CoNet packets. So, a reserved IP in IP destination field is used to distinguish between these two packets. Two other reserved IP is also used for the interest and data packets of CoNet [10, 17, 18]. A detailed part of the CoNet architecture is shown in Fig. 3.

On receipt of an interest to the edge node of the system, if the requested data isn't cached in any node, the interest will be routed to the appropriate destination which can be a system inner node in or edge node when the data belongs to an outer system. If the requested data is cached in a cache server, the interest will be routed to that server. When the data packet is routed to consumer, one of the switches will also send the data to be cached into a cache server. This should be done with the help of controller (NRS node).

The controller is also responsible for checking and ensuring that tags are unique in their domain. CoNet supports most of NDN's functionalities such as digital signature for security and in-network caching. It also offers some other functionalities such as TCP congestion control [10, 17, 18].

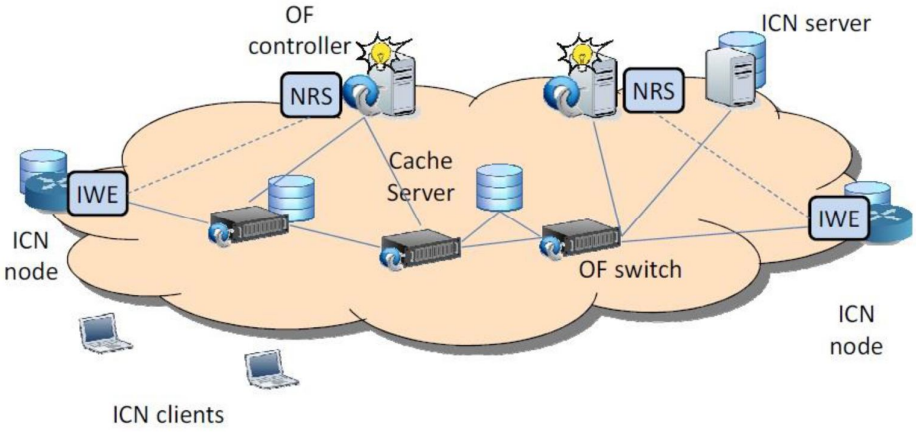

Fig. 3: CoNet architecture [17].

5. ADVANTAGES AND APPLICATIONS

In section 4 some methods for integrating SDN and NDN were discussed. In this section the advantages and applications of implementing NDN over SDN will be introduced. As mentioned before, one of the advantages of implementing NDN over SDN is to improve NDN's functionalities, such as TCP congestion control optimization in OFELIA. Also in [19], a routing protocol is proposed for supporting mobility in NDN by means of a controller. This can be easily implemented in integrated architectures of SDN and NDN. In [20] another controller-based routing strategy for NDN is proposed. The simulation results show improvement up to 75% signalling efficiency and faster convergence time. In [21], a routing scheme called the SDN-based Routing Scheme for CCN (SRCS) is proposed. The SRCS architecture is a clean slate implementation of NDN and uses NDN messaging, while it uses the separation of control plane and data plane in SDN. The simulation result on NS3 shows the improvement of caching efficiency and a decrease in the number of messages compared to the default flooding strategy in the original NDN architecture. Another case for improvement in NDN based on SDN is network and traffic management. In [22], a network management system is built on top of NDN by using SDN/OpenFlow controller to manage traffic by focusing on traffic classification, labelling, shaping and queuing.

Another advantage of combining NDN and SDN is to improve performance of new technologies such as virtualization, cloud computing, etc. For example, odICN [23] is proposed for data centers. OdICN is based on combining NDN and SDN/OpenFlow. In this framework, the users can request content from different data centers while the controller can optimize the downloading efficiency based on its global view of topology and resources. The odICN controller has three main responsibilities: content locating, content optimal deployment, and path optimization. Content locating can redirect users' requests to the best (least cost) data center containing the requested content. Content optimal deployment automatically deploys the contents to the other data centers according to the given principles for optimizing network performance; and path optimization and traffic balance are both done in spatial and temporal dimensions. In [24], the role of

virtualization, by use of SDN, in NDN is discussed. The authors outlined traffic optimization, traffic engineering, and caching management as the advantages of implementing NDN over SDN, especially for multimedia traffics.

In [25], an NDN architecture is proposed in the SDN framework for heterogeneous networks such as sensor networks. The motivation of combining these two networks in this paper is the flexible management and the ability for continuous evolution of network architectures in SDN, and also efficient data dissemination via name routing and in-network caching in NDN. A summary of mentioned advantages and applications is in Table 1.

Table 1: Application and Advantages of Implementing NDN using SDN

Paper	Application	Advantage
OFFELIA Testbed [17-18]	TCP congestion control	Optimizing TCP congestion control in NDN
J. Torres et al. [19]	Controller-based routing	Supporting for mobility in NDN
CRoS-NDN [20]	Controller-based routing	Signalling efficiency and faster convergence time in NDN
SRCS [21]	Controller-based routing	Caching efficiency, decrease in messaging
Q. Sun et al. [22]	Centralized traffic management in NDN	Higher awareness, fairness, and low congestion
M. Chen et al. [23]	Deploying NDN in data centers using OpenFlow	Obtaining content from data centers via its name, optimizing download efficiency in data centers, automatic and optimal deployment of content, load balancing in network links
J. Ren et al. [24]	Virtualization in NDN using SDN	Traffic optimization, traffic management, traffic engineering and cache management
S. Eum et al. [25]	Deploying NDN using SDN framework for heterogeneous networks such as sensor networks	Flexible management, efficient data dissemination, ability to evolve network incrementally

6. CONCLUSION

NDN is a new paradigm for dealing with various limitations of current network architecture and its applications. This new architecture focuses on “what” instead of “where”. It uses content naming and name-based routing instead of IP routing. But the current design of NDN still has some limitations and needs to grow. For exploiting and improving NDN functionalities, there is a need for dramatic changes to network devices deployed in current networks, which creates the challenge of NDN implementation.

SDN is another new network architecture that decouples the control plane from the data plane and places it into a server called the controller. One of SDN’s advantages and promise is the ability of deploying and improving new architectures and protocols without concerns for hardware dependency and expenses. So much research and papers have been done in implementing NDN using an SDN framework. This paper explained and discussed some of these methods. Advantages and applications of combining these new architectures

were also pointed out. The applications included traffic management, new routing schemes, virtualization, architectures for data centres and sensor networks.

REFERENCES

[1] Carvalho I, Faria F, Cerqueira E, Abelem A. (2012) ContentFlow: An introductory routing proposal for content centric networks using Openflow. API, 7th Think-Tank Meeting.

[2] Open Networking Foundation. (2012) Software-defined networking: The new norm for networks. ONF White Paper, available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, last accessed: 25 May 2015

[3] Zhang L, Estrin D, Burke J, Jacobson V, Thornton JD, Smetters DK, Zhang B, Tsudik G, Claffy K, Krioukov D, Wang L, Massey D, Papadopoulos C, Abdelzaher T, Crowley P, Yeh E. (2010) Named Data Networking (NDN) Project. avalilable online: http://named-data.net/publications/techreports/, last accessed: 25 May 2015.

[4] Rowshanrad S, Namvarasl S, Abdi V, Hajizadeh M, Keshtgary M. (2014) A survey on SDN, the future of networking. J. Adv. Comp. Sci. Tech., 3:232-248.

[5] OpenFlow Switch Consortium. (2009) OpenFlow Switch Specification Version 1.0. 0., available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, last accessed:25 May 2015

[6] Jacobson V, Burke J, Zhang B, Estrin D, Zhang L, Zhang B, Tsudik G, Claffy K, Krioukov D, Massey D, Papadopoulos C, Ohm P, Abdelzaher T, Shilton K, Wang L, Yeh E, Uzun E, Edens G, Crowley P. (2013) Named Data Networking (NDN) Project. available online: http://named-data.net/publications/techreports/, last accessed: 25 May 2015

[7] Nunes B, Mendonca M, Nguyen X, Obraczka K, Turletti T. (2014) A survey of software-defined networking: Past, present, and future of programmable networks. Communications Surveys & Tutorials, IEEE, 1617-1634.

[8] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. (2008) OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comp. Comm. Rev., 38: 69-74.

[9] Zhang L, Afanasyev A, Burke J, Jacobson V, Crowley P, Clafy K, Papadopoulos C, Wang L, Zhang B. (2014) Named data networking. ACM SIGCOMM Comp. Comm. Rev., 44:66-73.

[10] Salsano S, Blefari-Melazzi N, Detti A, Morabito G, Veltri L. (2013) Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed. Computer Networks, 57:3207-3221.

[11] Nguyen XN, Saucez D, Turletti T. (2013) Efficient caching in content-centric networks using OpenFlow. INFOCOM 2013 Student Workshop.

[12] Nguyen XN, Saucez D, Thierry T. (2013) Providing CCN functionalities over OpenFlow switches. Available online: https://hal.inria.fr/hal-00920554/, last accessed: 25 May 2015.

[13] Melazzi NB, Detti A, Pomposini M, Salsano S. (2012) Route discovery and caching: A way to improve the scalability of Information-Centric Networking. IEEE Global Communications Conference (GLOBECOM), 2701-2707.

[14] Melazzi NB, Detti A, Mazza G, Morabito G, Salsano S, Veltri L. (2012) An openflow-based testbed for information centric networking. Future Network & Mobile Summit (FutureNetw), 2012. IEEE, 1-9.

[15] Chanda A, Westphal C, Raychauldhuri D. (2013) Content based traffic engineering in software defined information centric networks. Proc. IEEE INFOCOM workshop NOMEN'13, 1-6.

[16] Ooka A, Ata S, Koide T, Shimonishi H, Murata M. (2013) OpenFlow-based content-centric networking architecture and router implementation. Future Network and Mobile Summit (FutureNetworkSummit), 2013. IEEE, 1-10

[17] Salsano S. (2013) OFELIA: EXOTIC final architecture and design. Technical report, available online: http://netgroup.uniroma2.it/people/faculties/stefano-salsano/stefano-salsano-publication-list/, last accessed: 25 May 2015

[18] Blefari-Mellazzi N, Detti A, Morabito G, Salsano S, Veltri L. (2013) Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed. Available online: http://netgroup.uniroma2.it/twiki/bin/view/Netgroup/CoNet, last accessed 25 May 2015

[19] Torres J, Ferraz L, Duarte O. (2012) Controller-based routing scheme for Named Data Network. Technical report, Electrical Engineering Program, COPPE/UFRJ.

[20] Torres J, Duarte O. (year?) CRoS-NDN: Controller-based routing strategy for Named Data Networking. available online: http://www.gta.ufrj.br/ftp/gta/TechReports/ToDu14a.pdf, last accessed 25 May 2015

[21] Aubry E, Silverston T, Chrisment I. (2015) SRSC: SDN-based routing scheme for CCN. IEEE NetSoft., Londres, United Kingdom.

[22] Sun Q, Wendong W, Hu Y, Que X, Xiangyang G. (2014) SDN-based autonomic CCN traffic management. Globecom Workshops,183-187.

[23] Chen M, Weng X, Wang X, Xing C, Zhang G. (2013) A mechanism of information-centric networking based on data centers. International Conference in Advanced Cloud and Big Data (CBD), 40-45.

[24] Ren J, Pentikousis K, Westphal C, Liu W, Wang J. (2013) The role of virtualization in information centric network deployment. IEEE COSMOC MMTC, 8:21-23.

[25] Eum S, Jibiki M, Murata M, Asaeda H, Nishinaga N. (2015) An ICN architecture within the framework of SDN. available online: http://www.ieice.org/~icn/wp-content/uploads/2015/04/ICN_kickoff_4.pdf, last accessed: 25 May 2015