Simulation Analysis of Characteristics and Application of Software-Defined Networks

Ivan Grgurević, Zvonko Kavran, Anthony Pušeljić (student)
Faculty of Transport and Traffic Sciences, University of Zagreb
Department of Information and Communication Traffic
Zagreb, Croatia
ivan.grgurevic@fpz.hr

Abstract - **Software-defined network (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of higher-level functionality. This research includes comparison of multiple scenarios of the software-defined network, which are based on different types of coverage and local area networks (LAN), i.e. a traditional LAN. Differences are evident in the scenario of network performance and can be perceived as advantages and disadvantages of SDN in relation to the traditional network. The parameters used in the analysis are data rate, packet delay (i.e. latency), packet loss, throughput, the cost of network performance and others. The application and the simulation demonstration of a software-defined network is shown in the graphical network simulator GNS and emulator Mininet. This research has analysed the advantages and disadvantages of a software-defined network over a conventional network, taking into account various parameters and stakeholders.**

Keywords - Software-defined network/networking, simulation analysis, controller, Application programming interface (API)

I. Introduction

Nowadays, we are witnessing a very high degree of application of virtualization technologies with the growing customer demand for a fast establishment and delivery of services and placement services within the Cloud Computing concept. In addition, users require flexible and automated network environment that is adaptable to current applicative requirements. Such new challenges require responses by the application of a different approach in relation to the classical network infrastructure management. Cloud computing allows users to store data and install software on the servers that are connected through the Internet. With the help of a web browser and special customers, these services are flexible and the users pay only for what they use.

Software-defined network (SDN) is a network architecture in which the networks control is separated from the packet forwarding and it contains the possibility of direct programming. Such migration of control, which is sometimes strongly related to an individual network device, in the external computing devices allows basic infrastructure separation of applications and network services, which are therefore able to treat the network as a logical or virtual entity. SDN enables dynamic adjustment of the network environment to the current application requirements or the user's needs, and simplifies management and increases the scalability of the network, which is particularly manifested through a simple implementation of additional network services and components. An additional benefit of SDN is the possibility of using the network components from different manufacturers, basically without having to know how to operate the devices since the complete network environment is managed from a single point, or through the SDN controller. The SDN network architecture consists of a controller SDN, OpenFlow network devices and a communication channel that connects them.

Today, the largest application of SDN is present in data centers which are also known as software-defined data centers (SDDC). Such data centers contain all the elements of the infrastructure needed for networking, storage, processing (Central processing unit - CPU), the realization of security and virtualization, and are being delivered as a service. Development, provisioning, configuring, and other operations of the whole infrastructure are separated from the hardware and executed by the software.

The aim of this research was to conduct an analysis of the characteristics and the application of software-defined networks. The analysis is based on a comparison of conventional networks and software-defined networks with the display of significant differences. The research includes a simulation of different network topologies using the graphical network simulator GNS3 on Linux. For the purposes of the simulation, it was necessary to specify the differences between network architecture of traditional networks and software-defined networks, and to conduct the process of designing software-defined network via a graphical network simulator GNS3 and emulator Mininet. The research is in fact an analysis of different scenarios and parameters (data rate, packet delay, packet loss, throughput, the cost of network performance, etc.).

II. Background and Related Work

Numerous available articles and research are dealing with the analysis of the characteristics and architecture of software-defined networks / networking, and the analysis is mainly based on the impact due to changes in certain performance of the network and the application of SDN controller [1], [2], [3]. Within the development of the Internet of Things (IoT) concept, many authors reveal the application of software-defined networks / networking and access in the IoT environment, and thus achieve the differentiation level of service due to the different needs of IoT in different (heterogeneous) scenarios, especially related to the wireless networks [4], [5], [6]. The development of software for the simulation of the operation of information and communications networks has achieved efficient testing of various networks and network elements, ways of networking and the presentation of various possible scenarios, which is also present within software-defined networks. Typically used software for the implementation of software-defined networks simulation is the OpenNet [7], Mininet [8], ns3 [9] and EstiNet [10]. The article [11] presents a comparative analysis of the existing simulators for SDN according to different characteristics and functions.

According to [12], the authors were interested in research of the SDN technology and its possibilities, and were thereby using Mininet simulator and POX SDN controllers. The results were compared with the results obtained by the application of network devices and the use of "traditional" network. The throughput in a software-defined network is increased in comparison to a “traditional" network and the number of lost packets in a software-defined network is smaller.

Within the SDN analysis conducted by the Open Network Foundation (ONF) it has been concluded that separate control and data planes result in better programmability, automation and better control of the network, which results in scalable and flexible networks which allow, for example, business companies to easily adapt to variable business needs [13]. Analyzing the issues of SDN, the Cisco Systems company has come to the conclusion that SDN greatly helps to simplify operations by automating and centralizing network business management [14]. One part of the research also analyzes the traffic parameters as part of the transport engineering in SDN networks, using various simulation methods and simulation experiments [15], [16].

III. OVERVIEW OF CHARACTERISTICS AND ARCHITECTURE OF SOFTWARE-DEFINED NETWORK

The SDN concept is based on the need to separate and redefine a network construction, and its implementation uses the following three principles:

1) Control and forwarding planes: Control planes are separated from the forwarding planes. Forwarding planes are still located in the switch, while control planes are moved to the SDN controller in the form of software.

2) Control intelligence: Control intelligence is centralized at SDN controller.

3) Network programmability by applications: The network can be programmed beginning from the applications. Applications interface can be exposed to the controller to manipulate the network.

The main objective of SDN is to achieve better management of networks with large extent and complexity and to ensure that all logical decisions of control level are made from the central point. This central access will reduce the need for the N-number of intelligent nodes in an N-nodes topology. The basic role of every network software is to program the path that will allow the traffic to flow. Now, when the dependence of software on the hardware is reduced, there is no need for intelligent software to operate on all nodes.

SDN is based on the concept of logical starting of software in a centralized location and programming of switches using the southbound Application Program Interfaces (API). Figure 1 shows the logical layers of SDN. At the lowest level there are network elements such as switches, computers, servers and other network devices. It is important to note that the switches are located on top of the lowest layer. The middle layer is a layer of controller that communicates with the switches.

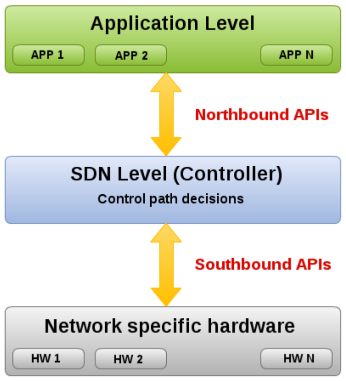


Figure 1 - Logic layers of SDN

The highest level is the application level in which the user can define the applications that will allow the definition of the network flow. As result, a network approaches the applications as one logic switch thus providing control of the entire network from one logic point and simplifies the network design and all of the operations within the network. SDN also simplifies the operating of the network devices because they no longer have to understand, but only to process a lot of protocol standards led only by the instructions of the SDN controller [14].

A. SDN Controller

The central controller (SDN controller) is a software entity that needs to have a global view on the entire network. The network operating system, launched logically for the choice of path, needs to be launched on the central SDN controller. The controller has an overview of the entire network and it can determine the optimized flow and program hardware ports. The basic characteristics of the controller are:

- Detection of end user devices such as laptops, desktops, printers, mobile terminal devices, etc.
- Detection of network devices that form the network infrastructure such as switches, routers and wireless access points.
- Management of network devices topology by maintaining information about the details of the link between the network devices and directly connected terminal devices.
- Control of database maintenance managed by the controller and performing of necessary coordination with the devices to ensure the synchronization of flow entry of devices with that database.

B. Southbound API

Within the architecture of software-defined network, the southbound API are being used for communication of SDN controllers with network switches and routers. Southbound APIs mitigate the efficient network control and allow the SDN controller to dynamically make changes according to the real-time requirements and needs.

C. *Northbound API*

Within the SDN network, northbound APIs are being used for communication of the SDN controller with the services and applications launched within the network. Northbound APIs can be used in order to mitigate the innovations and provide an efficient orchestration and automatization of network which can align due to its programmability with the needs of various applications. Northbound APIs are most critical of all within the SDN environment, because the value of SDN is related to innovative applications which can be potentially supported and provided and they have to support a wide range of applications.

IV. PLANNING OF SOFTWARE-DEFINED NETWORK

Many organizations inforce the initiative of the implementation of SDN solution, but there is a question of the best performance onto the more automated network architecture and what is to be considered and applied within. In many cases, the software-defined solution does not need to look any different from the conventional network. It is important to define the impact of the SDN model on the existing services and to use samples of applications that connect and checkout the continuity of the service before and after the implementation. That will prevent the disruption of service and eliminate all implementation-related problems. However, regardless of the number of preparations, some of the circumstances are still unpredictable. Therefore, it is important to have an alternative plan that allows the administrators to return the previous network configuration. The implementation of SDN without the proper knowledge represents a certain risk, but the ignoring of SDN represents a significant risk for IT organizations and IT experts. In the case of IT organizations, the risk is that they will not be able to solve the problems for which SDN has been designed, which results in the lack of competitiveness. The risk for IT professionals is that they can fall behind in learning and education related to this approach and thus will not have the competitive value for the current or for the future employer. SDN security needs to be built into the architecture, as well as delivered as a service to protect the availability, integrity, and privacy of all connected resources (and information).

According to previous analyses and research it can be concluded that in the upcoming period SDN will have a significant impact on corporate networks and roles of the network experts. Because of that, it is important that the IT organizations and the IT experts develop a plan for the SDN implementation. The implementation may vary depending on the size and the complexity of the network, as well as the experience of the IT team. New skills and additional training is needed. With proper planning, most organizations can quickly and easily take advantage of SDN solutions.

V. SIMULATION DEMONSTRATION OF SOFTWARE-DEFINED NETWORK OPERATION

A. *Development of the network topology*

Graphical Network Simulator 3 (GNS3) [17] was used for the purpose of creation of the conventional network topologies and the network configuration switches. It will be installed on the Linux OS, which allows the combination of virtual and real devices and allows the simulation of complex networks. It uses Dynamics emulation software to simulate Cisco's Internetwork Operating System (Cisco IOS). The software used in the configuration of the switches is used in the actual physical devices. The network topology used to compare the conventional configuration and the SDN network consists of five switches and two Linux PCs connected as shown in Figure 2.

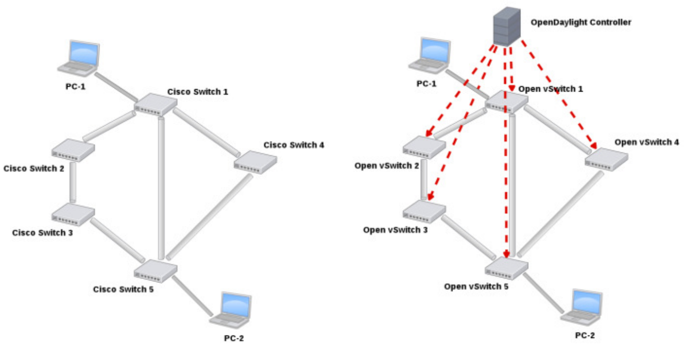


Figure 2 - Network topology of conventional network and SDN network [18]

By using GNS3 simulator, a network is created as shown in Figure 3. Before any traffic can flow from PC-1 to PC-2, it is necessary to configure all the switches, to make sure that the traffic from PC-1 to PC-2 flows via the shortest route, which is made possible by using FIB (Forwarding Information Base) on each switch.

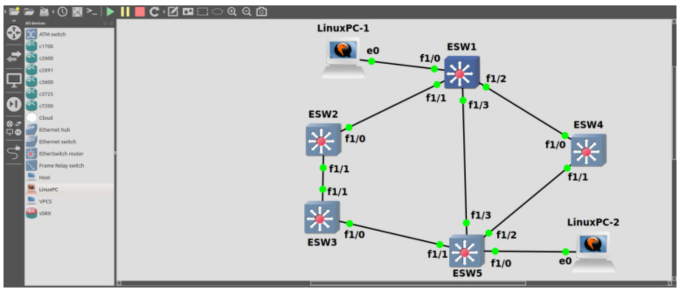


Figure 3 - Conventional network topology in GNS3 simulator

Once the ports, IP addresses and configured switches are assigned, the network topology must be learned by all switches.

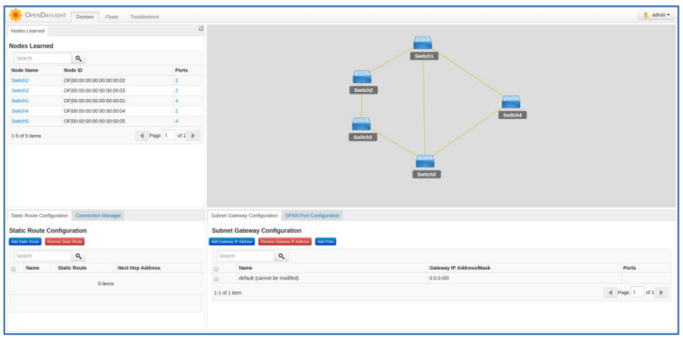
Even though the network topology contains a small number of network devices and is not complex, it can be concluded that the configuration of larger networks has higher demands for the number of connections and the time. If the network consists of a thousand network devices and hosts, which is very frequent in today's networks, every switch and every flow has to be particularly configured for the appropriate traffic. This process takes a number of procedures and increases the total time. Within the SDN solution all the procedures, from switch configuration and learning of network topology, are performed by SDN controller from one centralized point and within a very short period of time, which makes it an advantage over the conventional networks. The only condition is that switch has to be connected onto the SDN controller, and all the other work is performed by the controller itself.

The development of the SDN network topology is performed by Mininet [8] emulator in order to show how the controller operates. This emulator allows creation of virtual networks and initiates a real kernel, switch and the application code on virtual machine, which is in this case VM Virtual Box. Mininet is installed on the Linux OS and uses an appropriate script in *Python* programming language to initiate previously created network topology. The SDN controller is needed with the use of Mininet. For the purpose of testing OpenDaylight controller was used [19]. The topology consists of five Open vSwitches and two Linux PCs all connected as shown in Figure 2. After successful initiation of Mininet, it is necessary to start your own creation of topology specially intended for this case and written in Python programming language under the name of TestTopology. The command for initiation of the test network is:

sudo mn –mac –controller=remote,ip=192.168.165.1,port=663 –custom TestTopology.py –topo=mytopo, and the meaning of the individual parts of the command are:

- sudo mn: initiates command with *root* privilege
- --mac: sets MAC addresses of *hosts* similar to IP addresses, which makes it easier to read the generated traffic shown in Wireshark
- --controller=remote: informs the Mininet that SDN is not on the local computer
- Ip=192.168.165.1: IP address of the SDN controller, as well as the IP address of the *host* computer where the controller is started
- Port=6633: Standard TCP port for connecting the switch onto the controller
- --custom TestTopology.py –topo=mytopo: initiates its own topology written in *Python*.

By entering an accurate code, Mininet will create the network by adding controllers, hosts, switches and links that will configure the hosts and initiate switches. Figure 4 shows the visibility of switches and their connection in the OpenDaylight controller.



Nodes Learned

Node Name	Node ID	Ports
Switch2	OF|00:00:00:00:00:00:00:02	2
Switch3	OF|00:00:00:00:00:00:00:03	2
Switch1	OF|00:00:00:00:00:00:00:01	4
Switch4	OF|00:00:00:00:00:00:00:04	2
Switch5	OF|00:00:00:00:00:00:00:05	4

1-5 of 5 items Page 1 of 1

Static Route Configuration

Name	Static Route	Next Hop Address

0 items

Subnet Gateway Configuration

Name	Gateway IP Address/Mask	Ports
default (cannot be modifed)	0.0.0.0/0	

1-1 of 1 item Page 1 of 1

Figure 4 - The learned nodes in *OpenDaylight* controller

Even though this is about the creation of a virtual network, the used controller is also used in real physical networks. It is evident that this process facilitates the processes of a conventional network, where every single switch has to be configured manually, which is within the SDN controller quickly performed by separating the control planes from the data planes, which are still present within the switch. After the controller knows about the switch, the next step is to gain insight into the entire view of the network (i.e. learn about the details of switch devices and about the connections between them). This is conducted in two steps: the first step is to learn about the individual switches, and the second is to learn about the connections between the switches. The first step is performed by feature request and feature reply mechanisms. The controller sends feature-request message at the moment the so-called TCP handshake is conducted. The newly connected switch replies with the feature-reply message. The feature-reply message informs the controller about the capabilities of the switch, details of the port and the available operations. In the next step, the identification of the switch connections is made by *Link Layer Discovery Protocol (LLDP)* frames that are sent onto the connected ports of switches.

B. Performance measurement

Measuring of the performances includes two different scenarios, where the measurement of the permeability and packet delay within the client server communication based on the TCP protocol will be the first one, and the second one will measure the packet loss based on the UDP protocol. Scenario 2 is different in ending of the link triggered by closing the port between Switch 1 and Switch 5. Network topologies of conventional and SDN network are made in GNS3 simulator, to ensure the same conditions, on 4 Linux PCs by using real software with Cisco switches and Open vSwitch software based on SDN switches. Figure 5 shows the presentation of the used topology for Scenario 1.

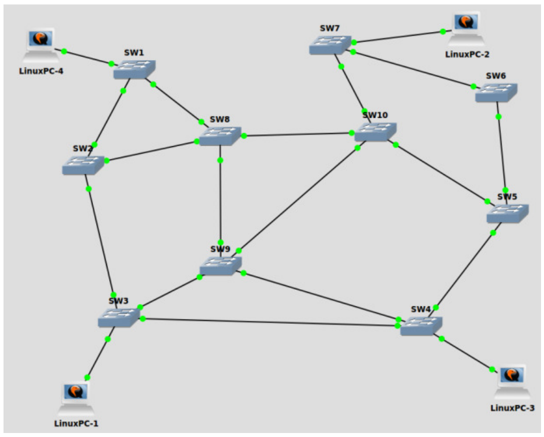


Figure 5 - Topology for Scenario 1

Conventional network is made of the following devices:

- 10 Cisco c3725 Ethernet Switch Router, and
- 4 Linux PC (PC-1 and PC-3 as client, PC-2 and PC-4 as server).

SDN network is made of the following devices and controllers:

- 10 Open vSwitch 1.11,

- 4 Linux PC (PC-1 and PC-3 as client, PC-2 and PC-4 as server), and
- OpenDaylight controller.

The traffic is generated within 5 minutes by the Distributed Internet Traffic Generator (D-ITG) [20] triggered on all PCs.

Before the measurement takes place, the necessary procedures are needed in order to enable the measurement, and these are:

1) To prevent deviations in measurement, all of the PCs are supposed to have their clocks synchronized and linked to the public Network Time Protocol (NTP) servers. For that reason, it is necessary to create gateways towards the Internet. That will be conducted by Cloud in GNS3 simulator, which is in fact a tunnel between the host computer and Linux PC. The IP address on the TAP host computer is also a default gateway for virtual Linux PC. In order for Linux to retrieve certain web addresses it is important to define the DNS server as the Google DNS server. The NTP server is a reference for the synchronization of the clock and that is, in this case, CARNET NTP server located in Zagreb (University Computing Centre of the University of Zagreb).

2) Creating of the Linux Bridge: Open vSwitch used in the SDN network is Virtual Box Appliance in the GNS3 simulator. Used ports must be added to Linux bridge so that the switches could communicate with OpenDaylight controller. After having performed successful needed configurations, the generator on PC-1 and PC-3 is initiated as sender, and the PC-2 and PC-4 as receiver.

The topology used for the presentation of Scenario 2 is shown in Figure 6. The settings are the same as inScenario 1, the only difference being the existence of one sender (PC-1) and one receiver (PC-2) and the use of UDP protocol.

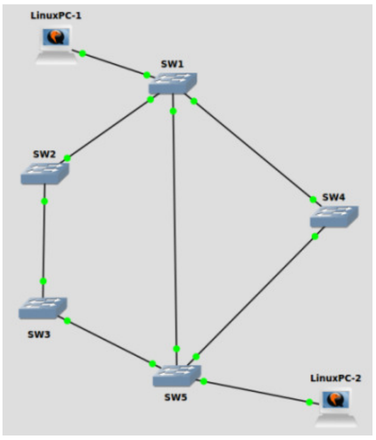


Figure 6 - Topology for Scenario 2

The measurement results are shown in Table 1 and Table 2. The results in the first measurement show that SDN offers equal performances if set conditions are identical. There is a slight difference in numbers because, since the topology is known in advance, the first packets in the beginning of the SDN solution will be sent faster than they would be in a conventional network. The difference is that the switches must first learn the topology and this creates the delay in relation to SDN. Although the conditions were identical and the used software real, these results in reality can be differentiated and vary because of influencing factors such as distance, links and, of course, the used hardware.

TABLE I. MEASUREMENT RESULTS FOR SCENARIO 1

Scenario 1		
Parameters	***Traditional network***	***SDN network***
Total packets	209,253	222,052
Avg. delay (s)	0.004798	0.004424
Bytes received	214,275,072	227,381,248
Avg. bitrate (Kbit/s)	5,714.098545	6,063.7873
Avg. packet rate (pkt/s)	697.521795	740.20841

TABLE II. MEASUREMENT RESULTS FOR SCENARIO 2

Scenario 2		
Parameters	***Traditional network***	***SDN network***
Total packets	265,425	285,875
Avg. delay (s)	0.002623	0.002791
Bytes received	271,795,200	323,456,000
Avg. bitrate (Kbit/s)	7,248.044165	8,625.693047
Avg. packet rate (pkt/s)	884.771016	952.941046
Packets dropped	59,920 (18.42%)	544 (0.19%)

It is very difficult to compare a traditional network with SDN on the basis of the measured performances because SDN is designed with the objective of flexible and easy network management. The SDN solutions are different depending on the manufacturers and the network can be configured according to the need. The performance is adjustable and it depends on the purpose of the organization and why it needs to use network services.

The second measurement shows higher difference in the results. While for SDN the loss was only 0.19%, for the conventional network it was 18.42%. The difference in reality can oscillate, but the SDN solution will definitely yield better results. Knowing the concept of the topology, the controller knows where to direct the packet if the link is disrupted or a certain port is closed, and its performance is very fast. In conventional switch devices, the topology has to be primarily learned because the switch operates only with the nearest unit and has no knowledge of the current state in the network. STP has four conditions, and these are: *blocking*, *listening*, *learning* and *forwarding*. Once the port is blocked it remains in that state for the next 20 seconds. Then it spends the next 15 seconds in the state of learning. If these two states are summed together with the *Hello* time of 2 seconds, the final time is 52 seconds. The difference in the measurement results is therefore high because the switches have to learn the topology, which is not necessary in case of SDN and this is an additional advantage.

VI. CONCLUSION

The software-defined network includes the architecture which can be described as dynamic, economic and adjustable which makes it ideal for the dynamic nature of today's applications. Separation of the control planes and data planes allows directly programmable network control and separation of the low-layer infrastructure for the purpose of applications and network services. SDN offers centralized view onto the network, providing the controller with SDN so that they can operate as control planes, which makes them a strategic control point within the SDN network. It communicates with switches/routers by using the southbound API, and with applications by using the northbound API. The centralized, programmable SDN environments are easily adjustable to the variable needs of the company. The key advantages of SDN are agility and flexibility due to its separated architecture. SDN allows the organizations to quickly develop new applications, services and infrastructures in order to satisfy the variable business goals, flexible selection and operation of the network. Implementing the SDN solution requires good planning. Organizations should have clear idea about the advantages that are planned to be achieved by implementing SDN. In many cases, software-defined solution does not have to look different from conventional network, and SDN solutions are different depending on the manufacturers.

During network configuration there are substantial differences that could be noted between SDN and the conventional network. In conventional network every switch must be configured separately, which requires more procedures and time. If the network consists of a thousand network devices and hosts, which is very common today, every switch has to be individually configured according to the current traffic flow and its changes, which additionally increases the number of procedures and the needed time.

Within the SDN solution, all of the listed procedures, from switch configuration and learning of new topology, are performed by SDN controller from one centralized point in the short time of the first connection. This concept is the main advantage of the SDN network in relation to conventional networks. The switch must be connected onto the SDN controller and the rest is performed by the controller itself. The advantage of learning of the entire topology and the view onto the entire network is shown on the basis of the results of Scenario 2 where the difference of the packet loss is high in regard to a conventional network.

REFERENCES

[1] B. Yeong Yoon, S.M. Kim, J.H. Lee: "Transport SDN Architecture for Distributed Cloud Services", The 12th International Conference on Optical Internet Proceedings (COIN), IEEE, Jeju, South Korea, 2014, pp, 1-2.

[2] S. Zhang, C. Kai, L. Song: "SDN based uniform network architecture for future wireless networks", Conference proceedings of International Conference on Computing Communication and Networking Technologies (ICCCNT 2014), Hefei, China, 2014, pp. 398-402.

[3] G. Sun, G. Liu, Y. Wang: "SDN architecture for cognitive radio networks", Conference proceedings of 1st International Workshop on Cognitive Cellular Systems (CCS), IEEE, Duisburg, Germany, 2014, pp 56-60.

[4] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, N. Venkatasubramanian: "A Software Defined Networking Architecture for the Internet-of-Things", Conference proceedings of Network Operations and Management Symposium (NOMS), IEEE, Krakow, Poland, 2014, pp. 1-9.

[5] V.R. Tadinada: "Software Defined Networking: Redefining the Future of Internet in IoT and Cloud Era", International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, Barcelona, Spain, 2014, pp. 296-301.

[6] H. Huang, J. Zhu, L. Zhang: "An SDN_based management framework for IoT devices", 25th IET Irish Signals & Systems Conference 2014 and China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014), IET Limerick, Ireland, 2014, pp. 175-179.

[7] OpenNet, available at: http://github.com/dlinknctu/OpenNet (25.06.2015)

[8] Mininet emulator, available at: http://www.mininet.org (17.06.2015)

[9] Ns-3, available at: https://www.nsnam.org/news/release-3-1/ (17.06.2015)

[10] EstiNet, available at: http://www.estinet.com/ (20.06.2015)

[11] M.C. Chan, J.X. Huang, T. Kuo, L-H. Yen, C-C Tseng: "OpenNet: A Simulator for Software-Defined Wireless Local Area Network", IEEE Wireless Communications and Networking Conference, Istanbul, Turkey, 2014, pp. 3332-3336.

[12] M.J. Todorović, N. D. Krajnović: "Simulation Analysis of SDN Network Capabilities", 21st Telecommunications forum TELFOR 2013, Serbia, Belgrade, 2013, pp. 38-41.

[13] ONF White Paper: "Software-Defined Networking: The New Norm for Networks", Open Networking Foundation, Palo Alto, CA, USA, 2012

[14] Cisco Systems: "Software-Defined Networking: Why We Like It and How We Are Building On It", White Paper, Cisco Systems, Inc., 2013

[15] S. Agarwal, M. Kodialam, T.V. Lakshman: "Traffic Engineering in Software Defined Networks", International Conference on Computer Communications (INFOCOM), Conference proceedings of INFOCOM, IEEE, Turin, Italy, 2013, pp. 2211-2219.

[16] M.R. Nascimento, C.E. Rothenberg, M.R. Salvador, C.N.A. Correa, S.C. De Lucena, M.F. Magalhaes: "Virtual Routers as a Service: the RouteFlow approach leveraging Software-Defined Networks", Proceedings of the 6th International Conference on Future Internet Technologies, New York, NY, USA, 2011, pp. 34-37.

[17] Graphical Network Simulator 3, available at: http://www.gns3.com (20.06.2015)

[18] A. Pušeljić: "Analysis of Characteristics and Application of Software Defined Networks", Master thesis, Faculty of Transport and Traffic Sciences, University of Zagreb, Zagreb, 2015, unpublished.

[19] OpenDaylight Controller, available at: http://www.opendaylight.org (02.07.2015)

[20] Distributed Internet Traffic Generator, available at: http://traffic.comics.unina.it/software/itg (25.06.2015)