ISSN:0975-9646

Performance Evaluation of Software Defined Networks

Habiba Amed[1], Taner Cevik[2,*]

[1]*Department of Computer Engineering, Istanbul Aydin University*

[2,*]*Department of Software Engineering, Istanbul Aydin University, Istanbul, Turke.y*

Abstract— **This paper aims to build an SDN which decouples the plane of data and plane of control. The implementation of the study is done using the virtualization technology. Virtualization technology is a networking technology which is used for the ease of application of networks without using physical devices. Thus, VMware created an interesting technology which allows a host operating system, like one of the popular Linux distributions, to run one or more client operating systems (such as windows). Therefore, in a flexible computing environment, the server is moved from one location to others and during the movement, they can pause the virtual machine and copy a file. Network virtualization is often mentioned as virtual private networks (VPNs) and occurs in various types. All of these theories are vital because they create the core inspiration for that is now called software-defined networking (SDN). Software-defined networks (SDN) is a design method which makes perfect and easy network operation. SDN has three logical layers, namely, application, control, and data. Besides, for communicating between the control and data planes, a protocol is used, that is called OpenFlow. OpenFlow is the origin of the decoupling of control and forwarding functions. According to the simulations that are conducted on the designed simple network architectures, the throughput of %90 is measured.**

Keywords— **Software Defined Networks, Data plane, Control plane, Application layer, OpenFlow.**

I. INTRODUCTION

Just a few years ago, network resources, storage and computing were deliberately separated operationally and physically. Even systems utilized to control these resources [1], [2], [3] are usually physically separated. Utilization that interacts by any of these resources, like an operational monitoring system that includes access policies, systems, and access procedures to a significant extent, is also at hand for security purposes. It is the preferred method of IT departments. Forcing organizations to combine these different elements occurs in data center environments immediately after the initiation (and demand) of cheap computing, storage, and networking. Applications that were introduced by this change of approach made the governance and running of these resources considerably closer than ever seen before.

Data centers [4] are outlined to physically separate conventional elements (e.g. Personal Computer servers), their associated repositories, and networks which connect them to the users. The power of computing in such data centers focuses on particular server functionality that runs utilization to serve desktop users, like mail servers, database servers, or other commonly used range of capabilities. Previously, these functions, which were typically deployed on desktops within an organization, were only used by departmental servers that provide services proper to local use. Department servers were moved to the data center as time went on for various reasons, such as ease of administration and sharing among users of the organization.

An interesting transformation took place about ten years ago. VMware companies made up an attractive technology which allows an operating system to run more operating system within the proper operating system [5]. VMware created a small program which can simulate all computer environment. Later, this technology brought together certain resources among virtual machines. The controller program has been named a hypervisor [6]. Initially, the design purpose is for those who need to run Linux in their programming requirements, or windows (it was an enterprise model at that time), designed for situations only that require the execution of a particular operating system environment. Due to the operating system virtualization's improvement, now the server can use a single, proper operating system like Microsoft Windows Server, and the specific platform is developed for that [7].

In a flexible environment, servers can be moved to any data center location and during movement, it can copy a file by pausing the virtual machine. With the improvement in technology, and supporting flexibility in-network, the cooling, and storage issue raise which will effect storage and computing power, and also operational efficiency [8]. Companies such as Amazon and Rackspace, which took up large amounts of storage and calculation for price efficiency, realized that they did not use all their computers and warehouses efficiently, and were able to resell their backup investment powers and warehouses to external users to compensate for investments.

On another method to overcome this dilemma is which during movement of the virtualization environments, performing environments are usually managed by the business organization. Generally, it is an ethernet LAN which connects virtual machines and all physical devices. In a multi-user data center, network resources, storage, and computing may be provided independently or within isolated segments. The main problem here is that the physical displacement means physical address displacement. When data centers are evaluated, network hardware stalled in terms of improvement further away speeds and feed. The Network operator can create a network and virtual network with using IP and MPLS [9], therefore data center operators can create virtual machines to execute virtual network instead of physical ones with using virtualization technologies. Virtualization of the network is often mentioned as virtual private networks (VPNs) and occurs on various types. When the idea of distributed plane of control reappeared.

Software-defined networking [10], is a technology which is based on these concepts and those concepts are vital, so they create core inspiration for that. Then it has been realized that this processing power can be used to logically operate a central control plane and use a potentially inexpensive commodity pricing switching equipment. Several engineers at Stanford University have formed an OpenFlow protocol that can be applied in such a configuration. OpenFlow is designed for several devices that contain only a plane of data for replying to command which is directed to them from a (logical) centralized controller that hosts a plane of control for this network. A hybrid approach is more likely where some of the networks are logically driven by a centralized control device, the other portions being operated by the more conventional dispersed control plane[8]. It should be noted which one of the inspirations to create SDN and OpenFlow is not only where it is programmed, but also the flexibility of how it can develop a network device.

Another example is the rapid programming of changes and then the provision of operational support systems (OSS) [11] implementations to provide fast and optimal access to the RIB. One of the most important issues around all these instances in which the discourse among the applications and the RIB is realized through the RIB manager.

II. Software-Defined Networks(SDN)

Although real or virtualized, it is a design method that makes perfect and facilitates network operations with closely linking the deal between applications. Generally, this is achieved using a centralized network control point - this is usually performed as an SDN controller - this then regulates, directs and simplifies the connection between applications that wish to operate with network elements and those that wish to communicate information to those applications. The control element then reveals, summarizes network functions and operations through new, application-friendly and bidirectional programmatic interfaces [8].

Software-defined, software-oriented and programmable networks have a rich and complex set of historical ancestry, objection and various solutions to these issues. This is the benefit of technologies that advance software-based, software-driven and programmable networks that advance technology based on what is possible. Although SDN controllers continued to run the press in 2013, many other developments took place at that time. One of the very interesting and bright ones is OpenDaylight. OpenDaylight aims to enable a group and open source framework that includes software and design to speed up and advance a common, powerful software-defined network platform [12].

A. *SDN Architecture*

The SDN architecture represents a new design which decouples the plane of control from data and facilitates network development, interoperability, and scalability. This separation is possible through the separation of key components, which are SDN "core". Besides, the fundamental differences between the "classical" network and the new networking paradigm are defined by three different logical layers as detailed below [13].

1) *Logical layers*: The SDN architecture can be represented by three different logical layers, as shown in Fig. 1.

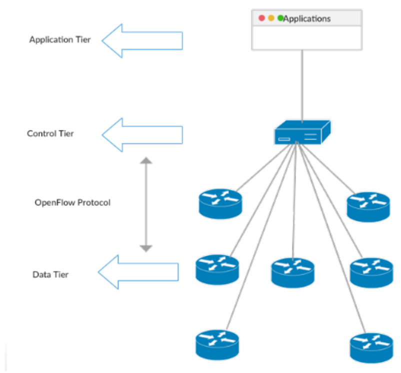

Fig.1 SDN Architecture

Each layer has different functions in particular:

a) The data plane layer shows the network infrastructure consisting of physical devices (e.g., switches and routers).
b) The controller layer represents "network intelligence" and is logically centralized in the SDN controller in this layer. This solution allows the controller to maintain the overall appearance of the network placed on the infrastructure layer.
c) The application layer represents the layer on which network operators and administrators can work. By centering the network state at the controller layer, it is possible to set up and control network resources using software-defined network programs in the application layer. Also, network operators can directly write and deploy customized programs without waiting for vendor versions that may take a long time.

The abstraction of the layers described above enables programmers to work on a network abstraction layer through Application Programming Interfaces (APIs) rather than working with thousands of different physical devices. However, this abstraction is only possible if the lower layer infrastructure makes it possible to interact with it[14].

A. *Control plane*

Describes the entity that is accountable of controlling forwarding traffic choices within the data plane. The controller is embodied as software that is enforced within the external central logic device. Therefore, it can be developed in the absence of the requirement for brand new hardware [15]. The control plane for the Internet is actually a compound of layer 2 and layer 3 control planes. Therefore, this ought not be surprising which the identical progress and development is needed for both layer 2 and layer 3 protocols and networks which form this portion of the networks.

The Layer 2 control plane concentrates on physical layer addresses, such as IEEE MAC addresses or hardware. A layer 3 control plane was created to ease network layer addresses like IP protocol addresses[16].

B. *Data plane*

The plane of data, processes the datagrams that are coming from wire or wireless media due to the link-level operations. It accumulates the datagrams and does basic accuracy/conformity testing. This is sometimes called a quick path to packet processing because there is no need to query other than defining the destination of the packet using pre-programmed FIB. The issue is for especially for the cases when packets may not confront to the standards, for instance, when a packet is received and it has an unknown destination, then this packet may be sent to the route processor. There, the control plane may perform more process on them utilizing RIB [17],[18].

A. *SDN switches*

On the one hand, thanks to the SDN architecture, the network becomes a "simple" packet forwarding element. Otherwise, routing decisions of high-level and status information are centralized in an external and separate server controller, rather than imposing policies and implementing protocols on the evolution of distributed devices expressed below[19].

B. *SDN Controllers*

An SDN controller provides services that transcend the concepts of transient management and centralization as well as realize a distributed control plane. A general description of the Software-defined network controller is:

- Distribution and management of the states are controlled by a database in some cases.
- It is a high-level data model which screens the relationships among managed policies, resources, and other services produced with the controller. Usually, these data models are constructed utilizing the Yang modeling language[20].
- The interface is ideally derived from the model which explains the controller's features and services. Some of the systems have more features and offer powerful environments, which permit the expansion of core modules, including those that promotes the dynamic expansion of controller features, and then release APIs for new modules:
- A secure TCP audit session between the controller and the associated agents in the network elements
- A standards-based protocol for providing network status for implementation in network elements
- A device, topology and service locator, a path calculation system, and potentially other network-centric or resource-centric information services[21].

a. Nox & Pox

Based on the Nox / Pox website [22], NOX was improved by Nicira and has become open source in 2008. NOX provides OpenFlow (OF v1.0) with a C ++ API and an asynchronous, event-based programming model. A truly significant advantage of widespread academic usage is the availability of a learning switch and sample code to emulate a network-wide switch that could be utilized as the starting code for different programming projects and experiments. SANE and Ethane are two famous NOX applications. Ethane is developed by Stanford University research group for network-wide security and centralized checklist level. Both have represented the effectiveness of SDN by significantly decreasing the lines of code that are needed to implement these functions, which in the past have received notably more code to execute similar functions. Due to its promising performance, researchers have shown MPLS applications on NOX core [23].

Pox Controller

POX is one of many controllers developed for SDN applications. POX is a sister project of NOX and has become faster than NOX. POX is a python-based OpenFlow controller. The main advantage of POX is that it is a Python-based Software Defined Networking controller for developing network applications. The reason we use POX is that it is already available and easy to learn and develop applications. Besides the opportunity of easy and free installation, PyPy works runtime, which enable it to work anywhere. POX is one of the controllers dedicated to create an archetypal, typical Software Defined Controller. ONOS, Floodlight, NOX, etc. Different SDN control devices are available. POX frames for both sides of OpenFlow; one for switch side and one for controller side [24].

The advantages of Pox has over Nox are listed below:

- Pox is a Pythonic OpenFlow interface.
- Pox possesses reusable sample components for path selection, topology discovery, etc.
- Pox works everywhere and comes with PyPy runtime, which does not require installation for easy development.
- Pox can be used in Linux, Mac OS, and Windows.
- It promotes POX, GUI and visualization tools in the same way as NOX.

- POX is working better when compared to NOX applications that are developed in Python.
- POX can communicate with NOX by the OpenFlow v1.0 switches and has specific promote for Open vSwitch[25].

C. *OpenFlow*

OpenFlow is a part of research at Stanford University. The main goal of this project is to allow the practice of protocols in campus networks, which can be utilized for research and experiments.

1) OpenFlow Protocol: OpenFlow is not a product of its own, but it is not one feature of a product; a group of protocols and an API. OpenFlow protocols are mainly portioned into two as follows:

- The protocols in the first group sets a wired protocol(version 1.3.x) to establish a control session, collect message structure(flow modes) and statistics to modify flow modifications, and define the fundamental structure of a switch.

- A configuration and management protocol based on NETCONF to allocate physical switch ports to the specific controller(currently version 1.1) that defines availability(active/stanby) and controller connection failure behavior[26].

2) OpenFlow Architecture: While OpenFlow fulfills a standard south-bound protocol (control element to the agent) to start streams, there is no standard for the north (application-facing) or east/west API. OpenFlow is one implementation of SDN, and indeed is the origin of the SDN because it splits data plane from the control plane. OpenFlow implements the data and control planes in distinct network elements. Fig. 2 shows the overall architectural view of OpenFlow. The network controllers are connected to the OpenFlow switches by using the OpenFlow protocol via a secure channel.

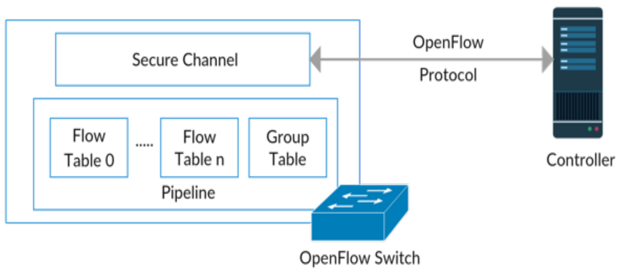

Fig. 2 OpenFlow Architecture

The Architecture Group seeks to handle, indirectly - defining a general SDN architecture. The critics arise about SDN architecture and OpenFlow are about the types of OpenFlow application services provided by an OpenFlow controller. They are not considered sufficient for all SDN applications[27].

D. SDN Weaknesses and Challenges

SDN and OpenFlow provide a way to simplify prototyping, distribution and management of network elements. However, there are some points to be considered that can make the network in an unsecured or unusable state:

- The usability of the controller is the main consideration.

- Security is also important. In SDN, the controller is critical component of the network, which exposes the controller to possible attacks and threats.

- The consistency of flow tables is also a potential issue. They may be subject to lower security checks, causing the flow tables to be in an inconsistent state.

- The scalability of the network depends on the controller, which is a potential bottleneck.

- The performance of the network can also be related to the adopted control model.

The issues described above have been considered by the community researchers and still stay as the future challenges for SDNs[28].

III. METHODOLOGY

In this section, an implementation of the SDNs is presented. In the outset, the environment for running the project is prepared. Hence, VirtualBox is installed, because macOS is used as the operating system. After that, we set up the Ubuntu operating system on VirtualBox and mininet on that.

A. *Ubuntu 18.04 LTS*

Ubuntu 18.04 introduced many new features as well as many minor changes. It is a long-term support release based on the Linux 4.4 kernel. Ubuntu repositories support many large desktops, including Ubuntu desktop (GNOME), K ubuntu (KDE), MATE Ubuntu, and GNOME Ubuntu (GNOME3). Ubuntu 18.04 has also featured like minimal installation. Instead of a complete set of applications, only the web browser and several utilities are installed[29].

B. *Mininet*

It is an emulator, which simulates a collection of network devices and connections in a Linux system. Lightweight virtualization is used to ensure that a single system executing the same system looks like a complete network. Mininet is vital for the open-source SDN group because it is often utilized as a simulation, verification, testing tool and resource. A mininet server acts as a real machine and usually runs the same code. A mininet server demonstrates the shell of a machine on which optional programs may be installed and run. These special programs may receive, send, and process using packets where a program looks like a real Ethernet, therefore, it is a virtual switch/interface. Virtual switches have processed the packets that as if a real switch or router, related to the method by which they are set up on Mininet hosts[30].

C. *Wireshark*

Wireshark is a network analyzer that reads, decodes and delivers packets on the network within an easy to understand format. Some features of the Wireshark are being open-source, active maintenance and the ability to degrade freely[31].

D. *Setting up the Pox Controller*

The Pox controller is adjusted in two ways:

1. It is first entered into the pox folder by typing cd pox/pox and then listing the existing files. Then the forwarding folder is entered using cd forwarding. The next step lists the files that are available in the forwarding folder. In the last step, the' sudo ~ / pox / pox.py forwarding.l2_learning' the command sets pox controller to learn[32].
2. The nano editor is accessed by using the 'su' command and system password. Then, the 'nano / usr / bin' command is used to enter the nano editor. After this stage in nano editor

'#!/bin/sh
Echo Pox "Controller Habiba Amed"
Sleep2'

commands are written. The pox controller is then set to debug the packets by typing:

Cd /home/habiba/pox && ./pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level --DEBUG[33].

After completing the adjustment process, we return to the command line. Then 'chmod a + x' is written on the command line. On Unix operating systems, the chmod command is utilized to change the access mode of a file. It stands for the change mode. 'a' is the abbreviation of all, which gives the users in the whole group access permission to the file. 'x' allows you to run the file or search the file if it is a directory[34]. Once the entire adjustment phase is complete, pox is started immediately after typing pox on the command line to test it.

E. *Designing an SDN in Miniedit*

Within the scope of this study, three types of SDNs are designed. They are single-controller SND, three-controller SDN and command-line-using designed SDN.

To design an SDN with Miniedit, firstly a terminal is opened from the application partition in the ubuntu operating system. The 'miniedit.py' file is then moved from the mininet folder to the home folder. Then, one controller, three switches and six host machines are added to the design.

In the next step, the controller is named as 'POX cont' and the controller type is selected as OVS controller. First switch is assigned an IP address 192.168.1.0. This process is repeated for the other switches respectively. Then all network devices are connected by clicking on the line tab. Then, IP addresses are assigned to the hosts. After this process is done on all six hosts, the first switch and the link of the first host is assigned the bandwidth of 100 MBits and the delay of 3ms. Fig. 5 demonstrates the designed single-controller SDN.

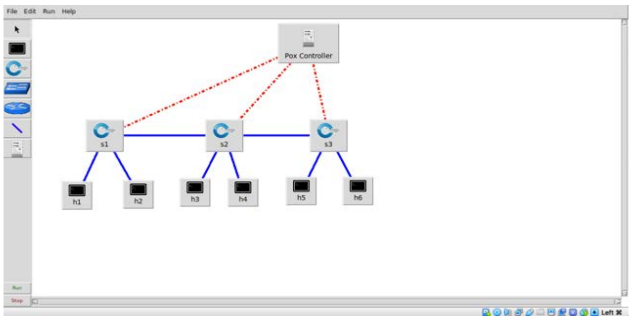

Fig.5 Single-controller SDN

The multi-controller software-defined network is designed in the same way and illustrated in Fig. 6.

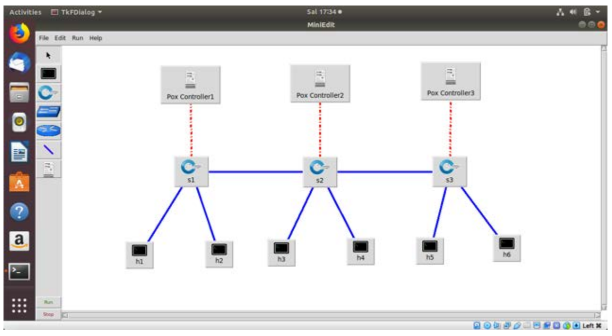

Fig.6 Multiple-controller SDN

To run both designs, you need to write the '.py' file on the command line. Before running both designs, it is necessary to start the Pox controller.

F. *Designing an SDN by Using the Command Line*

To design an SDN by the command line, first start the controller and then write the following command on the command line.

'Sudo mn --topo tree, depth=5,fanout=5 –controller = remote,ip=127.0.0.2,port=6634 –mac'

This command, designs a custom SDN of the depth five, branching three, a remote controller with the IP address 127.0.0.1. At the same time, the port number is assigned as 6633, as well as the network devices are assigned corresponding MAC addresses. This given IP is specific to the local controller.

IV. Simulation Results

The values obtained from the ping data are shown in Table 1.

TABLE I PING TIME RESULTS

Hosts	Packet Sent	Packet Received	Packet Loss	Minimum Time
h1-h40	7	7	0	0.077ms
h41-h80	7	7	0	0.084ms
h1-h68	7	7	0	0.077ms
h80-h120	7	7	0	0.07ms
h121-h161	7	7	0	0.077ms
h162-h200	7	7	0	0.092ms
h201-h243	7	7	0	0.081ms

As shown in Table 1, the number of packets sent and received across the entire network is equal. According to the ping test, the packet loss is zero and the periods differ.

V. CONCLUSION

Three types of networks are designed and tested. First type is the single-controller and the second one is three-controller and the last one is the network, which is created using the mininet command. The implementation is performed on MacOs by using virtual box and Ubuntu 18.04LTS. If these projects are applied on other operating systems, the results may differ. Besides, for the researchers that intend to design an SDN, either with commands or graphical user interfece, we strongly recommend them to use mininet with its miniedit software because of its easy deployment and implementation.

ACKNOWLEDGMENT

We thank to Ubuntu community for their technical support during the design of this research. We thank to stack Exchange website for their help during this research.

REFERENCES

[1] K. Goda, I. M. M. Kitsuregawa ve S. M. I. , «The History of Storage Systems,» *Proceedings of the IEEE,* cilt 100, p. 1437, 13 May 2012.

[2] *History of Computing.* [video]. 2017.

[3] S. Horiuchi, K. Akashi, M. Sato ve T. Kotani, «Network Resource Management Technology,» *NTT Technical Review, vol.*15, no. 10, pp. 2-3, Octobar 2017.

[4] H. Geng, «Data Center Overview and Strategic Planning,» *Data Center Handbook*, Palo Alto,CA,USA, John Wiley & Sons,Inc, 2015, p. 3.

[5] B. Ward, "VMware Virtual Machine," in *The Book of VMware—The Complete Guide to VMware Workstation*, K. Jurado , Ed., Canada, No Starch Press Inc., 2002, p. 8

[6] M. Parlakyigit, «parlakyigit,» 29 May 2013. [Online]. Available: https://www.parlakyigit.net/hypervisor-trleri/. [Accessed: 24 September 2019].

[7] M. Minasi, C. Anderson, B. M. Smith ve D. Toombs, «Windows 2000 Server Overview,» *Mstering windows 2000 Server,* Third Edition., P. Gaughan ve C. Henry, Dü, Alameda,CA, SYBEX,Inc., 2001, pp. 1-3.

[8] Seyir Defteri, İTÜBİBD:BİLGİ İŞLEM DAİRE BAŞKANLIĞI,7 September 2013. [Online]. Available: http://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/06/mpls-(multi-protocol-label-switching---%C3%A7oklu-protokol-etiket-anahtalama). [Accessed: 24 September 2019].

[9] A. Contini, «Opendaylight,» 16 November 2016. [Online]. Available: https://www.opendaylight.org/blog/2016/11/16/software-defined-networking-fundamentals-part-1-intro-to-networking-planes. [Accessed: 30 September 2019].

[10] H. Hanrahan, «Operation Support Systems,» *Network Convergence : Services,Applications,Transport, and Operations Support*, John Wiley & Sons, Ltd, 2007, pp. 385-398.

[11] T. D. Nadeau ve K. Gray, «Introduction,»*SDN: Software Defined Networks: an authoritative review of network programmability technologies.*, Sebastopol, CA, O'Reilly Media, Inc., 2013, pp. 1,2,3,4,5,7,8.

[12] A. Sonba ve H. Abdalkreim, «Performance Comparison Of the state of the art Openflow Controllers,» Halmstad, 2014.

[13] B. N. Shaker, «Software Defined Networking Architecture and Design based on Energy Conserving Model,» Baghdad, 2017.

[14] G. R. de Tejada Muntaner, «Evaluation of OpenFlow Controllers» 2012.

[15] A. M. MohamedAhmed, A. Mohamed Musa, M. A.-E. Ahmed ve M. W.-t. Alameen, «Designing Dynamic Consistency for Multi-Controller Software Defined Network Topologies,» Hartum, Sudan, 2017.

[16] I. Ahmad, S. Namal, M. Ylianttila, S. M. I. A. Gurtov, S. M. ve I. , «Security in Software Defined Networks: A Survey,» *IEEE Communications Surveys & Tutorials, vol.*17, no. 4, pp. 2317-2346, 27 August 2015.

[17] P. Charalampos, «A study on Software Defined Networks:Traffic Engineering,» 2013-2015.

[18] P. Goransson ve C. Black, «SDN Controller,» *Software Defined Networks*, MA 02451, USA, Elsevier Inc., 2014, pp. 68-72.

[19] tootoonchian, «noxrepo/nox,» GitHub, 14 February 2014. [Online]. Available: https://github.com/noxrepo/nox. [Accessed: 9 September 2019].

[20] S. Harsh, «Evaluation of Multiple Controller based Software Defined Networks Architecture over Single Controller Software Defined Architecture,» 2016.

[21] T. D. Nadeau ve K. Gray, «SDN Controllers,*SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies*, M. Loukides ve M. Blanchette, Sebastopol, CA 95472, O'Reilly Media, Inc.,, 2013, p. 89.

[22] N. C. Fernandes ve L. C. S. Magalhes, «Control and Management Software for SDNs:Conceptual Models and Practical View,*Network Innovation through OpenFlow and SDN: Principles and Design*, 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742, U.S.A: CRC Press:Taylor & Francis Group, 2014, pp. 88-92.

[23] R. Petersen, «Ubuntu 18.04 Introduction,» *Ubuntu 18.04 LTS Desktop: Applications and Administration*, Alameda, Surfing Turtle Press, 2018, pp. 1-8.

[24] C. Patras, «A study on Software Define Networks,» Piraeus, 2013-2015.

[25] F. ONGARO, «ENHANCING QUALITY OF SERVICE IN SOFTWARE-DEFINED NETWORKS,» Bologna, 2013–2014.

[26] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown ve S. Shenker, «NOX: Towards an Operating System for Networks,» *ACM SIGCOMM Computer Communication Review, vol.* 38, no. 3, pp. 105-110, July 2008.

[27] F. ONGARO, «ENHANCING QUALITY OF SERVICE IN SOFTWARE-DEFINED NETWORKS,» Bologna BO, İtalya, 2013-2014.

[28] F. Hu, Editor ve M. Farooq, «Language and Programming in SDN /OpenFlow,» *Network innovation through OpenFlow and SDN: principles and design.*, 6000 Broken Sound Parkway NW,Suite 300 Boca Raton, FL 33487-2742, U.S.A: CRC Press Taylor & Francis Group, 2014, pp. 76-77.

[29] F. ONGARO, «ENHANCING QUALITY OF SERVICE IN SOFTWARE-DEFINED NETWORKS,» Bologna, 2014.

[30] A. Orebaugh, G. Ramirez, J. Burke, L. Pesce, J. Wright ve G. Morris, «Introducing Wireshark:Network Protocol Analyzer,» *Wireshark & Ethereal Network Protocol Analyzer Toolkit*, Rockland, Syngress Publishing, 2007, pp. 52-53.

[31] M. Team, «Mininet,» Octopress, 2018. [Online]. Available: http://mininet.org/walkthrough/. [Accessed: 16 September 2019].

[32] y. a. pratama, Director, *Mininet Part 2.* [video]. 2017.

[33] M. I. KHAN, «GeeksforGeeks,» GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/chmod-command-linux/. [Accessed: 16 September 2019].