SIMULATION OF SOFTWARE DEFINED NETWORKS WITH OPEN NETWORK OPERATING SYSTEM AND MININET

Antonio Cortes

Department of Computer Engineering, University of Panama, Panama City, Panama

ABSTRACT

With the emergence of recent technologies in the field of computer network, traditional infrastructure in the field of networks have become obsolete and incompatible with respect to the new architectures of open networks that emerge with force. This is how software-defined networks emerge by enabling cloud computing ecosystem, enterprise data centers, and telecommunications service providers. The major contribution of this paper is the simulation of an ecosystem based on a software defined network by making use of certain types of networks topologies and using the virtualization of the open network operating system (ONOS) and Mininet as a network emulator.

KEYWORDS

Computer Network, Software-defined Networks, Network Topologies, Simulation, Open Network Operating System.

1. INTRODUCTION

The infrastructure of the traditional networks was developed and implemented in such a way the flow control and the routing oversaw the devices of the network, thus allowing static structure [3], hierarchical and dependent on the network architecture. This makes the network a complex scenario, because there is to accommodate the structure of the network to the needs of users, taking into consideration the policies of a network [1] and the growth of these, since they are problems those who face these designs.

Currently, with the emergence of social networks, smart devices and cloud computing, the various network topologies that make up these infrastructures tend to become saturated and consume a large bandwidth due to the electronic components of these innovative technologies. In turn, the benefits of cloud computing and virtual storage are being limited by networks since the implementation of new networks prevents them from meeting the new needs demanded by them and demands from the operators' market. In turn, the above gives rise to the emergence of a new network model that improves the capabilities in these networks called Software Defined Networks (SDN) [2].

In this way, Software Defined Networks are a network design architecture in which the control layer and the data layer within a network are separated. It separates the hardware management software from the network and transfers the control to other devices called controllers that convert the data traffic and control the network into a centralized service [4]. At the same time, the emergence of these networks is to cover a need related to the deficiencies of current networks

DOI: 10.5121/ijcsit.2018.10503

because, by reducing the hardware necessary for assembly, facilitates the reuse of hardware by facilitating the management of network control elements and makes it simpler in its configuration process and reduces management time for administrators, speeding up the deployment of applications, services and infrastructures.

Therefore, SND networks allow to ensure that network engineers and administrators respond quickly to changes in business by centralizing the control console, by improving the services on the network by making them more dynamic, economical and scalable avoiding management at a low level. Similarly, SDN networks are composed of three essential parts. On the one hand, we have the controller that oversees managing the network and telling the rest of the devices how to handle the traffic on the network. The Southbound APIs is a software like the OpenFlow protocol, responsible for managing the communication between the controller and the devices. While the Northbound APIs is responsible for establishing communication with applications and business logic. In this way, for a network to be functional it is required that the devices of this have incorporated a firmware with OpenFlow or another similar protocol. This means that SDN network flexibly manage the devices through the controller, which is responsible for recognizing the topology of the network, thus enabling better management of traffic loads in a flexible and efficient manner, giving priority to processes, applications and services.

In this paper, we have considered the use of the Open Network Operating System (ONOS) and the Mininet network emulator to be able to simulate the Software Defined Network (SDN) ecosystem and their respective network topologies. It makes use of a graphical interface for the virtual machine VirtualBox of Oracle, version 5.2.8 r121009 (Qt 5.6.2) in which ONOS is installed with its respective Mininet emulator, the OpenFlow controller and the Ubuntu Operating System 16.0.4.3.

The organization of this paper is as follows. In Section 2. Methods and Materials refers to the inputs used in the preparation of this paper. In turn, Section 3. Results and Discussion, the analysis of SDN is illustrated through ONOS and Mininet simulation tool. Section 4. Explains the final considerations and details the references used in this paper.

2. METHODS AND MATERIALS

A systemic process is carried out that allows the ONOS, Mininet and revision of the documentations from the various positions proposed by each one of the different authors.

2.1. OPEN NETWORK OPERATING SYSTEM (ONOS)

The Open Network Operating System is an open source distributed SDN control platform, developed by the Open Networking Lab (ON, Lab) [5], and sponsored by some of the leading companies and academic institutions. In comparison with Open-DayLight [6], the ONOS Project is specifically oriented to ISP (Internet Service Provider) networks, facilitating high availability and scalability, due to its distributed architecture. The identification of the network topologies, as well as our results obtained in the SDN simulations, are carried out using the version of ONOS 1.13.1 and the Mininet network emulator.

On the other hand, to establish the design of the SDN network, as well as the number of controllers or data switches, number of switches, identification of physical links through which the exchanged packets travel between the control plane at the application level, not only the ONOS platform was used, but two complementary approaches were also adapted. On the one hand, we use the SDN Mininet network emulator [7], to create a test network topology, which is shown in Figure 1., and we proceed to execute an instance of ONOS with fwd (Simple reactive

forwarding application), application responsible of generate traffic between the nodes of the network through the Mininet network emulator. Through a network snnifer, we analyse the data flow due to OF messages exchanged between each emulated switch and ONOS. It must be considered that a port of a node can be connected to many nodes, for example, through switch / hub.

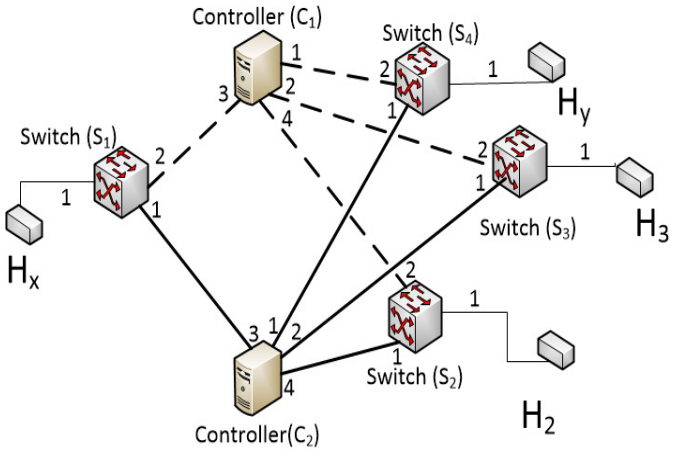

Figure 1. Example of network that connects H_x and H_y through a route P_{xy} = 4 switches. Dashed lines indicate the communication between the switches and the ONOS controller

In turn, to validate the interaction in the control plane and generalize our findings to any network, we analyzed in detail the source code of ONOS, and the number of flow rules was evaluated based on the number of flow packets received. Both reactive applications in ONOS work in layer 2 and assume a single IP address. The data flow is identified by a pair of source and destination addresses, respectively, both at the MAC level and at the IP level. However, a data stream may comprise multiple sessions at higher protocol levels, for example, TCP, UDP or RIP. Inside a swith, we find a port connected to another swith and it is declared as an internal port, while a port connected to a node, client and/or server, is identified as a host port. The network topology that interconnects the switches, that is; the internal ports and the communication links between them are previously known by the controller, thanks to the preliminary phase of topology discovery based on the LLDP [8] protocol implemented by the controller sending specific pkt_out messages to the switches. In turn, the location of the host, is given from the port where the host is connected, is disconnected a priori so the controller must locate it in real time.

2.2. ONOS WORKFLOW

We consider that the flow configuration process establishes a bidirectional communication between a source node (H_x) and a destination node (H_y), assuming that the ports of the corresponding nodes are unknown to the controller (C_1 or C_2). Let P_{sd} be the number of switches along the shortest route from H_x to H_y. Let N be the total number of controllers in the network, and let E be the number of communication links between the hubs or switches. We evaluate the exact number of devices, cluster nodes, nodes, links, packet processor, partitions, ports (bits per second), port statistics (packets per second) and data flow statistics (bytes), for the fwd application. For simplicity, we refer to the tree topology shown in Figure 1. , as a reference case, but our results generally apply to any given network topology. We assume that the ARP tables of both H_x, H_y and the forwarding tables of all the switches are initially empty.

2.3. MININET

An emulator is software that allows programs to be run on a different platform than the one originally designed. Unlike a simulator, it only reproduces the behavior of the program while an emulator accurately models a device that can be compared with the original hardware.

MiniNet [9] is one of the first emulators explicitly developed to support SDN, by allowing the efficient execution of small-scale networks with artificial traffic on computers that are not necessarily powerful, its license is free and permissive (BSD - Berkeley Software Distribution).
In addition, implementing the network with a large number of network devices is very difficult and expensive. Therefore, to overcome these problems, the virtual mode strategy has been carried out in order to create prototypes and emulation of technological networks using the MiniNet emulator. Its operation is carried out through a single Linux kernel and uses virtualization in order to emulate a complete network using only a single system. However, the node created, as well as the switches, routers and links are real elements although they are created by software [10].

The goal of MiniNet is to create virtual networks, running nodes, network cores and virtualized network devices simply and quickly through a simple feature host, with an open and free environment such as Linux. In turn, it has the ability to emulate different types of elements of the network, such as: nodes, layer 2 switches, layer 3 routers and links.
Some features that led to the creation of MiniNet are:

- Flexibility, that is, new topologies and new features can be configured through the use of software, by implementing common programming languages and operating systems.
- Applicability, allows correct applications made in prototypes. They should also be able to be used in real networks based on hardware without any change in the source codes.
- Interactivity, responsible for managing and executing the simulation of the network, so it must occur in real time as if it were happening with a real network.
- Scalability, prototyping must be scaled in large networks with hundreds or thousands of switches on a single computer.
- Realistically, the behavior of the prototype must represent the real behavior of time with a high degree of confidence, so the application and protocol stacks should be usable without any code modification.
- Shareable, prototypes created should be easily shared with other collaborators, who can then execute and modify the elements [11].

2.3.1 MININET WORKFLOW

Mininet has the capabilities that allow researchers or network programmers to create a new software-defined network in a prototype and simple way, with the ability to interact, customize and share, and provide a way to be executed on the hardware.

- Creating a network.

You can create a network in MININET with a single command;

$ sudo mm --switch = ovsk, protocols = OpenFlow13 --controller = remote --topo = tree, depth = 4, fanout = 2 --ipbase = 172.17.0.2 / 20

Create a virtual network of four switches or switches connected in a tree topology to two hubs

and 20 nodes, each node configured for the corresponding switch, so there would be a distribution of 5 nodes per switch, as show in Figure 2.

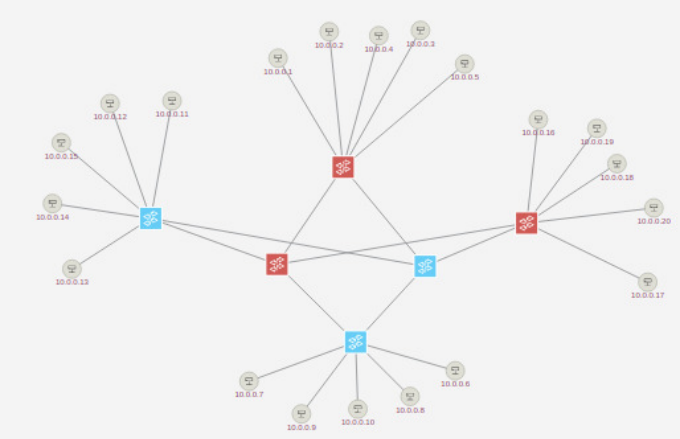

Figure 2. Network topology in Tree and its components

- **Interacting with a network**

In Mininet, the entire virtual network can be controlled, and managed from a single console, for example, the CLI command.
Mininet> h1 ping -c3 h2
It is used to send a ping to node h2 from node h1.
Mininet> nodes
View the list of available nodes.
Mininet> help
Allows you to see a list of available commands.
Dpctl: controls and edits flow tables.
Iperf: Test the TCP speed.

- **Customize a network.**

Custom networks with a few lines of Python can be created with the Mininet API. For example, ~ / mininet / custom / topo-4sw-20host.py

These few lines of Python create a virtual network of twenty nodes connected through virtual links to four switches.

- **Share a network.**

Mininet allows you to share the created, an image of VM to other researchers with the purpose of running, evaluating or modifying it.

3. RESULTS AND DISCUSSION

Initially, the Open Network Operating System (ONOS) was activated, through the icon named Setup ONOS Cluster. Subsequently, the network topology configuration called Spine Leaf Topology or tree network topology is activated in such a way that activates all the components of this topology. At the same time, the virtualization process of the

network is accessed through the ONOS GUI icon, which allows the researcher to observe the behavior of the network with all its elements in real time.

Once we have activated the virtual ecosystem of the network, we proceed to identify the existing devices in the network topology, as shown in Table 1.

Table 1. Devices.

Device	Device Id	Master	Port	H/W Version	S/W Version	Protocol
Spine-1	of:0000000000000001	172.17.0.3	5	Open vSwitch	2.5.4	OF_13
Spine-2	of:0000000000000002	172.17.0.4	5	Open vSwitch	2.5.4	OF_13
Leaf-1	of:000000000000000b	172.17.0.4	8	Open vSwitch	2.5.4	OF_13
Leaf-2	of:000000000000000c	172.17.0.3	8	Open vSwitch	2.5.4	OF_13
Leaf-3	of:000000000000000d	172.17.0.3	8	Open vSwitch	2.5.4	OF_13
Leaf-4	of:000000000000000e	172.17.0.4	8	Open vSwitch	2.5.4	OF_13

As shown in Table 1. , a total of 6 devices have been identified, of which Spine-1 and Spine-2 refer to controllers C_1 and C_2, respectively. On the other hand, the Leaf - 1, Leaf - 2, Leaf - 3 and Leaf - 4, are the switches that are part of the tree network topology.

Also, we can observe that each device has assigned a binary identifier of 16 digits, where the last digit is a number and in another it is a letter. This is done in order to differentiate the controllers of the switches in the network. Similarly, we have identified the hardware version that is the Open vSwitch, the software version 2.5.4 and the communication protocol that is OF_13. The master IP address 172.17.0.3, assigned to C_1 allows communication to be established with S_2 and S_3, respectively, while 172.17.0.4, which is the IP address configured in C_2, communicates S_1 and S_4.

In turn, we also identify the total of links that are 8 and that are part of the network, as shown in Table 2.

Table 2. Link.

Port 1	Port 2	Type	Direction
of:0000000000000002/1	of:000000000000000b/2	Direct	A ↔ B
of:000000000000000b/1	of:0000000000000001/1	Direct	A ↔ B
of:000000000000000c/1	of:0000000000000001/2	Direct	A ↔ B
of:000000000000000c/2	of:0000000000000002/2	Direct	A ↔ B
of:000000000000000d/1	of:0000000000000001/3	Direct	A ↔ B
of:000000000000000d/2	of:0000000000000002/3	Direct	A ↔ B
of:000000000000000e/1	of:0000000000000001/4	Direct	A ↔ B
of:000000000000000e/2	of:0000000000000002/4	Direct	A ↔ B

In Table 2. , we observe the respective ports 1 and 2, the type of link assigned to each port, which is direct and the direction taken by the data flow in the network that is full-

duplex, represented by the equation A↔ B, which means that the data bits, zeros and ones, travel simultaneously between two or more devices.

In Table 3. , we identify the total of nodes or host that are 20, with the following parameters.

Table 3. Host.

Host	Host Id	Mac Address	Location
10.0.0.1	00:00:00:00:00:01 / None	00:00:00:00:00:01	of:000000000000000b/3
10.0.0.2	00:00:00:00:00:02 / None	00:00:00:00:00:02	of:000000000000000b/4
10.0.0.3	00:00:00:00:00:03/ None	00:00:00:00:00:03	of:000000000000000b/5
10.0.0.4	00:00:00:00:00:04 / None	00:00:00:00:00:04	of:000000000000000b/6
10.0.0.5	00:00:00:00:00:05 / None	00:00:00:00:00:05	of:000000000000000b/7
10.0.0.6	00:00:00:00:00:06 / None	00:00:00:00:00:06	of:000000000000000c/3
10.0.0.7	00:00:00:00:00:07 / None	00:00:00:00:00:07	of:000000000000000c/4
10.0.0.8	00:00:00:00:00:08 / None	00:00:00:00:00:08	of:000000000000000c/5
10.0.0.9	00:00:00:00:00:09 / None	00:00:00:00:00:09	of:000000000000000c/6
10.0.0.10	00:00:00:00:00:0A / None	00:00:00:00:00:0A	of:000000000000000c/7
10.0.0.11	00:00:00:00:00:0B / None	00:00:00:00:00:0B	of:000000000000000d/3
10.0.0.12	00:00:00:00:00:0C / None	00:00:00:00:00:0C	of:000000000000000d/4
10.0.0.13	00:00:00:00:00:0D / None	00:00:00:00:00:0D	of:000000000000000d/5
10.0.0.14	00:00:00:00:00:0E / None	00:00:00:00:00:0E	of:000000000000000d/6
10.0.0.15	00:00:00:00:00:0F / None	00:00:00:00:00:0F	of:000000000000000d/7
10.0.0.16	00:00:00:00:00:10 / None	00:00:00:00:00:10	of:000000000000000e/3
10.0.0.17	00:00:00:00:00:11 / None	00:00:00:00:00:11	of:000000000000000e/4
10.0.0.18	00:00:00:00:00:12 / None	00:00:00:00:00:12	of:000000000000000e/5
10.0.0.19	00:00:00:00:00:13 / None	00:00:00:00:00:13	of:000000000000000e/6
10.0.0.20	00:00:00:00:00:14 / None	00:00:00:00:00:14	of:000000000000000e/7

In Table 3. , we observe the IP addresses assigned to each node, as well as the respective identifier and mac address associated with each node. In the range of IP addresses 10.0.0.10 to 10.0.0.15, hexadecimal values are assigned to both the identifier and the mac address of the respective nodes. In the location, each IP address of each node has the respective switch assigned by a letter and the node number.

On the other hand, in table 4. , we obtain the processing of packets at different times of activation of the topology of the network.

Table 4. Packets Processors.

Priority	Packets					Average (MS)				
	P1	P2	P3	P4	P5	A1	A2	A3	A4	A5
Priority 0	623	36	540	676	1576	6.47826	30.17557	4.45899	3.73259	2.00214
Priority 1	623	36	540	676	1576	4.42091	0.09236	4.54083	3.63926	1.57453
Priority 1	623	36	540	676	1576	1.31084	0.00300	1.08449	0.86684	0.37290

In Table 4. , we can see that packet processing will have three priority types, 0, 1 and 1, respectively, established for this tree network topology. We did tests in the network with different time intervals of connection and disconnection of the same, to observe the behavior at the level of the processing of packages and the average of them. In addition, it was possible to observe the

bits per second at the ports level, the packets per second at the level of the statistics of the ports and bytes at the level of the data flow statistics.

In Figure 3a.,and Figure 3b. , we can observe the relationship between the different priorities with respect to the packages and the averages, respectively.

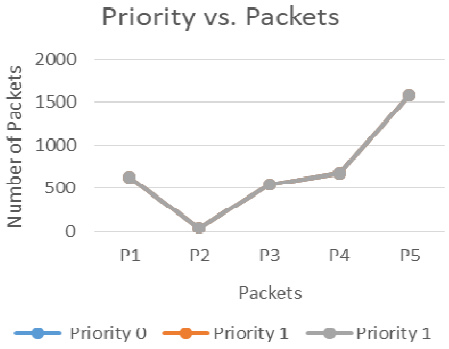

Figure 3a. Priority vs Packets

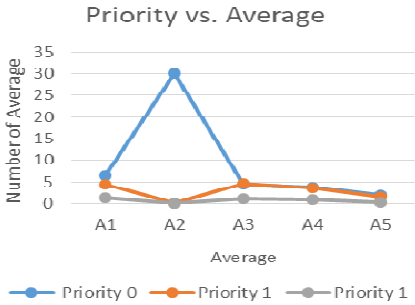

Figure 3b. Priority vs. Average

In Figure 3a. , we can see that at the level of priority 1, in gray, the maximum number of packets sent in the network reaches its maximum trajectory in 1576 packets, while the lowest is in 36 packets. Between P3 and P4, the oscillation tends to increase with a difference of 136 packets ((P4 = 676) - (P3 = 540) = 136). In the case of Figure 3b. , it is observed that the maximum average is reached when A2 = 30.17557 milliseconds (ms) in priority 0, in blue. The lowest average is given when A2 = 0.00300 milliseconds (ms) in priority 1, in gray. The latter is due to a decrease in the bandwidth of the network or otherwise a disconnection in some of its links due to a fault has occurred.

In the same way, other performance metrics were taken into account to evaluate the performance of the network. These parameters are related to the bandwidth which is analyzed through each of the ports located in the controllers and switches, respectively, as we can see in Table 5.

Table 5. Ports for controllers and switches.

Device	Enabled	ID	Speed	Type
Spine-1	False	Local	0	Copper
	True	1	10000	Copper
Spine-2	True	2	10000	Copper
	True	3	10000	Copper
	True	4	10000	Copper
Leaf-1	False	Local	0	Copper
	True	1	10000	Copper
Leaf-2	True	2	10000	Copper
	True	3	10000	Copper
Leaf-3	True	4	10000	Copper
	True	5	10000	Copper
Leaf-4	True	6	10000	Copper
	True	7	10000	Copper

In table 5. , we can see that both for the controllers, Spine-1 and Spine-2, and for the switches, Leaf-1 to Leaf-4, the medium used for data transmission is copper and bandwidth is 1 Mbits per second or 10,000 transmission bits in the infrastructure of the links. It is also observed through the ID, which ports are active, in local status, true or false, both for the controllers and for the switches.

In turn, another performance metric to evaluate is throughput, which is evaluated through the ports per device of those packets that are sent and received, both at the switch and controller level through the communications network.

In the case, of the switches, identified as Leaf-1 to Leaf-4, which you have assigned 8 port, 3 flows and 0 tunnel, you have for the switch, identified as: of: 000000000000000b, the following information as observed in Table 6.

Table 6. Port for Device (Switches)

Port ID	PKTS Rx	PKTS Tx	BYTES Rx	BYTES Tx	PKTS Rx DROPPED	PRTS Tx DROPPED	DURATION (SEC)
1	341	362	38416	40070	0	0	753
2	327	360	37156	40252	0	0	753
3	12	365	962	40564	0	0	753
4	12	377	962	41252	0	0	753
5	12	377	962	42344	0	0	753
6	14	377	1122	42910	0	0	753
7	12	377	962	42910	0	0	753

In Table 6, we can observe that the throughput is analyzed from the received and sent packets in a network through a communication channel. The number of packets sent is greater than that of the received ones. Similarly, bytes sent with respect to received bytes happen. No packets sent and transmitted dropped. All this has an estimated 753 seconds.

On the other hand, for the controllers, Spine - 1 and Spine - 2 have 5 ports of which 4 are used, since one is local. It also has 3 flows and 0 tunnel. In Table 7, we can observe, specifically in the Spine - 2, the sent and transmitted packets.

Table 7. Port for Device (Controllers)

Port ID	PKTS Rx	PKTS Tx	BYTES Rx	BYTES Tx	PKTS Rx DROPPED	PRTS Tx DROPPED	DURATION (SEC)
1	693	709	66113	70383	0	0	1218
2	711	705	69129	70575	0	0	1218
3	711	711	69675	70935	0	0	1218
4	705	709	69119	71115	0	0	1218

As with the switches, we can see that the transmitted packets are larger than the received packets. In turn, the bytes transmitted are greater than the bytes received. There are no dropped shipping and receiving packages. All this has an estimated 1218 seconds.

However, and with all the above, the network topology proposed in this work is compared with the tree topology with the MPLS topology (Multiprotocol label switching), since it is a very popular method used for traffic control and the creation of virtual private networks (VPNs). This method known as "tunnel-less", is a bit complicated to understand because it lacks a point-to-point connection. In Figure 4, the MPLS topology is shown.

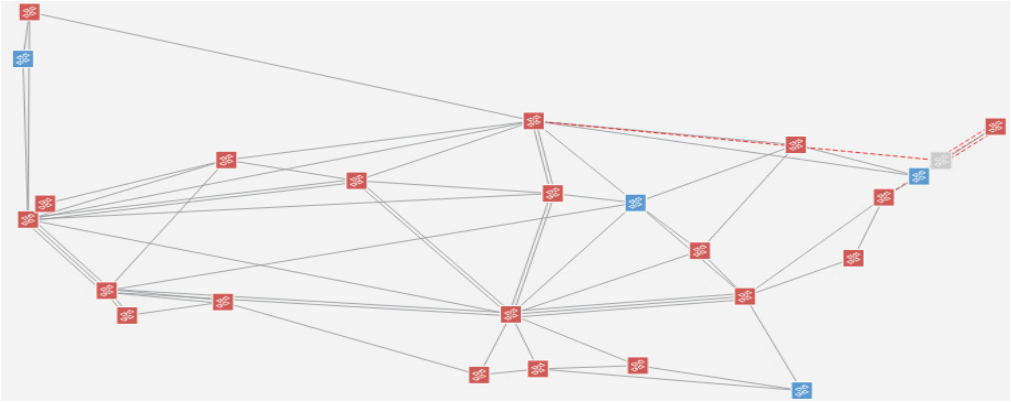

Figure 4. MPLS Topology

This scenario, which is presented in Figure 4, allows us to observe that at the level of communications infrastructure, what dominates the most are switches interconnected with fiber optic links or, ultimately, copper. It is important to mention that in this type of topology there is a drop in logical links due to the creation of virtual private networks. The existence of subnets composed of nodes is not unlike the tree topology that does exist. In this type of scenario, clients connect to the backbone of the network through multiservice links, which provides high transport speed and connectivity.

In addition, it is important to highlight the data packet flow that occurs in this type of topology with respect to the tree topology, as shown in Figure 5a-b.

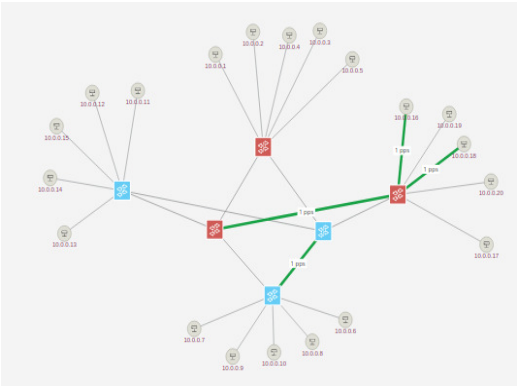

Figure 5a. Flow packets in Tree Topology

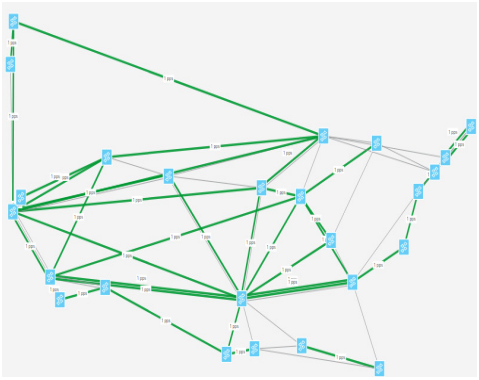

Figure 5b. Flow packets in MPLS Topology

It is observed, in Figure 5a. and 5b. , respectively, in green, the flow of data packets is more intense in the MPLS topology than in the Tree topology. Other reasons, which may

lead to the fall of the logical links in the MPLS topology is the means by which the data is routed since it is copper.

4. CONCLUSIONS

In this article it is proved that the Open Network Operating System (ONOS) and the Mininet network emulator allow researchers in the field of networks to virtualize a network topology in tree and any other, with all its main actors as are: the switches, the controllers, the nodes connected to each switch, establish the IP address for the network in general and the IP addresses for each subnet, among other relevant aspects. Similarly, it could be observed in the tree network topology, when the links between different nodes break or fall or between the controller and a node. This situation is represented in the virtual network, as discontinuous lines in red. It was observed how the bits per second are transmitted between the various ports of each device. This process of data transmission was reflected in the network in green.

On the other hand, a comparison process is carried out at the level of new network performance metrics between which the bandwidth and throughput are considered. Both metrics are analyzed taking into account the ports enabled in each switch and controller, packets received and transmitted at the level of these devices. Also, a comparison is made at the level of the proposed network topology against the MPLS topology, among which the most outstanding is the flow of data packets between both topologies and that the MPLS topology does not have sub-networks formed by nodes but interconnections between different linked switches by means of a communication channel formed by copper or optical fiber.

It is important to point out that where the experiments were carried out, the Hardware and the Software used were installed in a desktop-like technological infrastructure. We used a Lenovo Laptop with a 64-bit Operating System (Windows 10), an Intel Celeron 3205 U processor with a speed of 1.50 GHz and 4.00 GB RAM. In the same way, it was installed and made use of a virtual machine, Oracle VirtualBox, where ONOS and Mininet were installed, thus optimizing the infrastructure and improving the performance of those elements that were available at that time.

As future lines of research, ONOS would be used with other types of network emulators to analyze the behavior of different topologies of more extensive networks and to start with a comparison process, thus allowing us to observe and compare new results in this type of ecosystem.

REFERENCES

[1] Carvalho, L. Fernando, Abrão. T, Mendes L. de Souza, ProençaJr. L. Mario, An Ecosystem for Anomaly Detection and Mitigation in Software-defined Networking, Expert Systems with Applications (2018), doi: 10.1016/j.eswa.2018.03.027

[2] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, Kuala Lumpur, 2015, pp. 205-210, doi: 10.1109/ISMS.2015.46

[3] Hartpence. B and Rosario. R, Software Defined Networking for Systems and Network AdministrationPrograms, The USENIX Journal of Education in System Administration, November 2016, Volume 2, Number 1, https://www.usenix.org/jesa/0201/hartpence.

[4] Schaller. S and Hood. D, Software Defined Networking Architecture Standardization, Computer Standards & Interfaces, (2016), doi:10.1016/j.csi.2017.01.005

[5] ONOS Controller, URL https://www.onosproject.org.

[6] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S.Banergee, Devoflow: scaling flow management for high-performance networks, SIGCOMM Comput. Commun. Rev.41 (4) (2011) 254-265

[7] Mininet Network Emulator, (Available at http://www.mininet.org).

[8] LLDP Link Layer Discovery Protocol, (IEEE Standard 802.1 AB).

[9] J. Metzler y S. Taylor, «Network World, » 08 06 2011. [En línea]. Available http://www.networkworld.com/article/2177684/lan-wan/the-growth-in-eastwest-traffic.html.

[10] Mininet. An Instant Virtual Network on your Laptop.2014, Accessed: Sept. 2014[Online] Available: http://mininet.org.

[11] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: rapid prototyping for software-defined networks,” in Proceedings of the 9thACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.

Authors

Antonio Cortes Castillo is a computer engineer trained in the Latin American University of Science and Technology (ULACIT), Costa Rica, 1995. He obtained his bachelor´s degree in computer engineering with an emphasis in Management Information System at the Nacional University of Heredia, Costa Rica, 2002. Adquiere his Master´s degree in Computer Science (Telematics) at the Technological Institute of Costa Rica. He is now teaching at the University of Panama and is with his PhD at the University of Alicante.