Academic Journal of Research and Scientific Publishing | Vol 4 | Issue 45
Publication Date: 5-1-2023 ISSN: 2706-6495

Implementation of Hub, Switch and Load Balancer Scenarios in a Software-Defined Datacenter Network

Obay Mohammed Hassan Abadi
Student, Department of Computer Systems and Networks, Sudan University of Science and Technology, Khartoum, Sudan
Email: obay601@outlook.com
Khaled Fath-Alrhman Algzole
Student, Department of Computer Systems and Networks, Sudan University of Science and Technology, Khartoum, Sudan
Dr. Niemah Izzeldin Osman
Associate Professor, Department of Computer Systems and Networks, Sudan University of Science and Technology, Khartoum, Sudan
Email: niema.osman@gmail.com

Abstract:

Software-Defined Networking (SDN) is a networking design approach, which separates the data plane from the control plane, providing the network with certain benefits due to centralized programmatic control. SDN provides a centralized view and enables management of the entire network. It offers flexibility in configuration, reduces time to deploy, provides automation and facilitates building a network without requiring the knowledge of any vendor-specific software/hardware. SDN offers the datacenter network benefits such as improving traffic and security management as well as providing a scalable infrastructure. Load balancing enhances the performance of the network by distributing traffic among servers and therefore preventing congestion and server underutilization. This paper aims to explore SDN and test its implementation and demonstrate how to implement its concepts within a datacenter network. The experiment considers the datacenter network implemented in Sudan University of Science and Technology as a case study. The edited network topology is made up of a Demilitarize Zone (DMZ) having three servers, a Local Area Network (LAN) and a main switch. The Hub, Switch and Load balancer scenarios are implemented using Mininet emulator.

In addition, the three scenarios are tested using external devices and analyzed using Wireshark. Moreover, the connectivity of HTTP and FTP is verified.

Keywords: Control Plane, Data plane, Load Balancer, Network Programmability, Software-Defined Networking.

1. Introduction

Software-Defined Networking (SDN) is a recent architectural approach that improves and simplifies network processes by combining interaction such as provisioning and messaging among applications, services and devices, whether they are real or virtualized. SDN is deployed by employing a logically centralized control point in charge of coordination and management of communication between applications and network elements that require interaction with each other. In addition, the controller recognizes and identifies network processes. The controller then reveals network functions and processes through application-friendly and bidirectional programmatic interfaces 0(Feamster et al., 2013) .

SDN allows the deployment of devices from different vendors in a flexible manner. Employing standard protocols instead of vendor proprietary protocols enhances the flexibility of the network in terms of interoperability between access, distribution and core layers that consist of different vendors, the price factor in purchasing the devices and support.

2. Research Objectives:

The objectives of this research are to:

1. Propose a load balancer for SDN-based datacenter networks.
2. Testing the load balancer under the Hub, Switch and Load balancer scenarios using both emulation and external devices.
3. Testing the connectivity of HTTP and FTP services in the SDN network.

3. Research Importance:

Datacenter networks are designed for satisfying the data transmission demand of densely interconnected hosts in the datacenter, so the adaptation of load balancing methods became common and important. However, the requirement of load balancing routing in datacenter networks cannot be fully satisfied by traditional approaches.

4. Related Work:

A number of studies on SDN have been proposed (Mahjoub, 2015), (Körner , 2015) (Salih et al., 2014) (Ghaarinejad, 2015). The project Event-Driven Network Control Using Software-Defined Networking explored SDN as an emerging paradigm and tested its implementation in dynamic network environments (Mahjoub, 2015). It also highlighted the problem of dynamic networks in terms of configurability and the need to look at SDN as an approach or architecture to not only simplify the network but also make it more reactive to the requirements of workload and services placed on the network. The work in (Körner , 2015) dealt with the question of how datacenter networks could benefit from the integration of SDN in terms of network primitives, load balancing, Quality-of-Service (QoS) overlays and forwarding and firewalls. It also introduced innovative concepts to realize the usual ingress datacenter network service chain. The authors of (Salih et al., 2014) took advantages of SDN and implemented a remote configuration of Virtual Local Area Network (VLAN) from a single controlling point. The case study was to balance the load between multiple VLANs when one of them is overloaded. The work in (Ghaarinejad, 2015) emphasized the need for an SDN load balancer. In traditional networking, the IP address is used to split traffic among servers. Therefore, the traffic is not equally split, causing congestion and utilization problems. The problem was solved using SDN and three different load balancing policies were developed: Round-Robin, Random and Load-Based.

The algorithm proposed in (Chakravarthy & Amutha, 2019) balances traffic load in data center networks by selecting the best paths to route data flows. Paths are evaluated based on: byte rate, packet count, forward rate and transmission duration. Experimental results proved that the algorithm relieved network congestion due to load balancing which lead to reducing latency. The load balancer implemented in (Aguilar & Batista, 2019) employed three algorithms: random, round robin and least bandwidth. Results showed that the performance of the three algorithms is similar, maintaining acceptable response time. A SDN Multiple Controller Load-Balancing Strategy Based on Response Time (SMCLBRT) was proposed in (Cui et al., 2018) considering the trade-off between load balancing and switch migration among controllers.

The evaluation revealed that switch migration could be performed ahead of time to offload congested controllers while keeping the number of migrating switches minimum.

The load balancing problem was addressed in (Sufiev et al., 2019) by dynamically reassigning switches among controller clusters. Simulation results indicated a fivefold improvement of dynamic clustering over static clustering. Another study attempted to manage load balancing in hierarchical networks by distributing traffic flows among different paths in the network (Sitota, 2021). The proposed load balancing algorithm transmits traffic on the path having the least cost considering throughput and delay. Simulation results proved that the proposed algorithm reduced congestion while improving resource utilization. The authors of (Mulya et al., 2019) used ant colony optimization for load balancing in a SDN network. A simulation was developed to evaluate completion time, throughput and traffic load. Attained results showed that the ant colony optimization resulted in higher throughput compared to round robin while ensuring even distribution of CPU utilization.

5. Software-Defined Networking:

The Open Networking Foundation (ONF) has formulated the definition of Software-Defined Networking (SDN) as an innovative network architecture that separates network control from forwarding and that it is directly programmable (ONF, 2012). According to this definition, SDN is characterized by two properties; the separation of the control plane and data plane and enabling programmability of the control plane. It is worth mentioning that these two features are not new to network architecture, however, SDN offers both features with higher programmability, flexibility and control.

5.1. SDN Architecture

Fig.1 depicts the SDN architecture. As the figure shows, there are three different layers (Xia et al., 2015):

1) ***Application Layer:*** Provides several services and applications like access control, intrusion detection and prevention system (IDS/IPS), in addition to load balancers.
2) ***Control-plane Layer***: Includes a logically-centralized SDN controller, which maintains a global view of the network that takes requests through clearly defined Application Programming Interfaces (APIs) from the application layer and performs management and monitoring of network devices via standard protocols.
3) ***Data-plane Layer***: Also known as infrastructure layer, this layer provides with hardware components used for forwarding such as switches and routers.

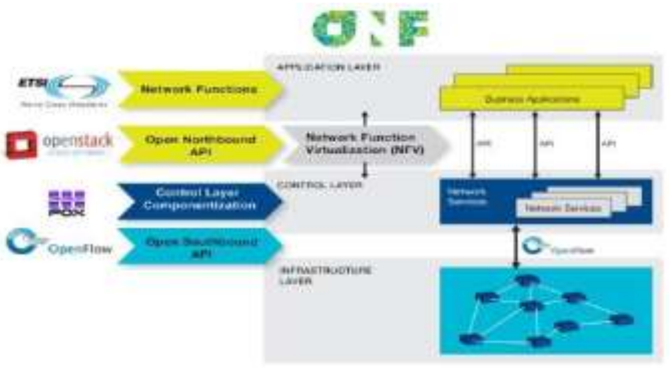


Fig.1. The Open SDN Architecture

5.2. SDN Controllers

SDN controllers differ in their basic architecture and programming model in addition to other concepts. Finding the ideal controller for a specific situation is influenced by many factors including the choice of programming language which can sometimes affect performance and the learning curve of the controller. Other factors include user base and community support. Moreover, the choice of controller should consider the Southbound and Northbound interfaces (policy layer).

There are several popular SDN controllers implemented in different languages offering a wide variety of services. The most well know controllers are summarized in TABLE **1**.

TABLE 1 SUMMARY OF PROPERTIES OF FAMOUS CONTROLLERS

	NOX	**POX**	**Ryu**	**Floodlight**	**OpenDay Light**
Language	C++	Python	Python	Java	Java
Performance	Fast	Slow	Slow	Fast	Fast
Distributed	No	No	Yes	Yes	Yes
OpenFlow version	1.0	1.0	1.0, 1.1, 1.3, 1.4	1.0	1.0, 1.3
Learning Curve	Moderate	Easy	Moderate	Steep	Steep

5.3. Load Balancing

Today's Internet experiences high traffic that could lead to server congestion if not managed in a controlled manner. Load balancing is the mechanism of distributing traffic among servers with the purpose of offloading them leading to improved performance. It is achieved using Application Delivery Controllers (ADCs) which direct traffic to a set of servers hosting the same target application. Load balancing ensures a stable, falt-tolarent network with improved performance (Ghosh & Manoranjitham, 2018).

As illustrated in Fig.2, the load balancer device resides between the clients and the set of servers. It accepts incoming traffic and distributes it to servers using a load balancing algorithm. This reduces the load on each server. In the case of a server failure, the load balancer redirects packets to other responsive servers.

A load balancing algorithm defines the method that is used to select the server to which each client request is forwarded. Load balancing algorithms differ in the criteria they use for this selection. The most famous load balancing algorithms are (Load Balancing Algorithms, 2021):

1. The Least Connection Method
2. The Round Robin Method
3. Weighted Round Robin
4. Fastest Response

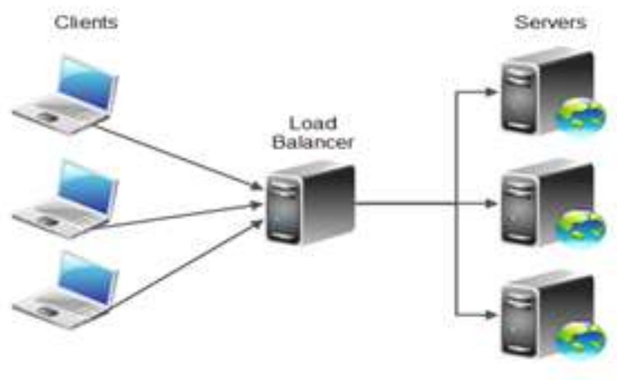


Fig.2. Load Balancer

5.4. SDN for Datacenters

A datacenter is a centralized repository, either physical or virtual. It employs many host servers and networking devices that process requests. In addition, it interconnects hosts to other hosts in the network or to the public Internet. Requests made to a datacenter range from web content serving, email and distributed computation to many cloud-based applications. It simultaneously provides many applications associated with a publicly visible IP address. Fig.3 shows the hierarchical topology of a datacenter network.

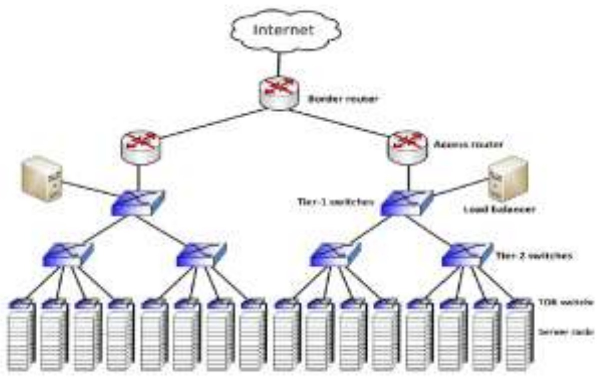


Fig.3. The Hierarchical Topology of a Datacenter Network

6. The SDN Design and Emulation Environment

6.1. Network Topology

In this study, Sudan University of Science and Technology (SUST) datacenter network topology was selected to implement SDN.

SUST network contains one main core switch connecting the LAN network with the Demilitarize Zone (DMZ) area. Inside the DMZ there is one server with multiple services provided to the network as shown in Fig.4. This topology was edited in order to give a clear implementation of SDN concepts and to meet load balancing main expectation which is distributing traffic to more than one server. The edited topology contains three basic legs: The DMZ area which contains three servers, LAN and the main switch.

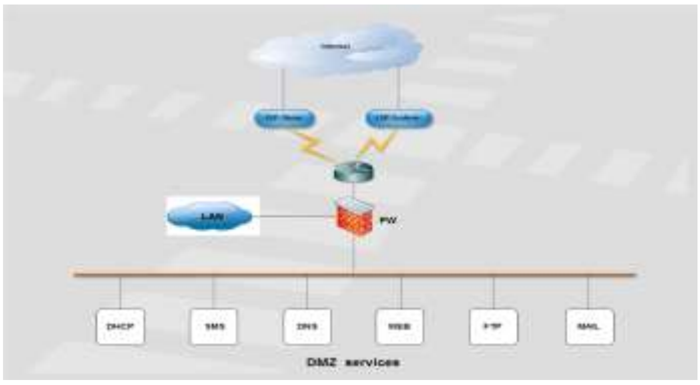


Fig.4. SUST Datacenter Network Topology

6.2. The Emulation Environment

Mininet emulator (Mininet Overview, 2019) was chosen for this experiment because it combines many of the best features of emulators, hardware testbeds and simulators. Compared to full system virtualization-based approaches, Mininet:

- Boots faster: seconds instead of minutes.
- Scales larger: hundreds of hosts and switches vs. single digits.
- Provides more bandwidth: typically, 2 Gbps total bandwidth on modest hardware.
- Easily installed: a prepackaged Virtual Machine (VM) that runs on VMware or VirtualBox for Mac/Win/Linux is available with OpenFlow v1.0 tools already installed.

Compared to hardware test beds, Mininet is inexpensive, always available and quickly reconfigurable and restartable. Compared to simulators, Mininet runs real, unmodified code including application code, OS kernel code and control plane code (both OpenFlow controller code and Open vSwitch code).

6.3. Controller Scenarios

According to TABLE 1, there are many SDN controllers available to implement OpenFlow protocol to control OpenFlow devices and act as the aggregated control plane. The comparison between these controllers led to choosing POX as the SDN controller in this work.

In the considered topology there is one Open vSwitch (OVS) to direct traffic to the servers. And so, the controller will manage the switch to work as desired. Three scenarios are implemented and tested on the OVS in the network:

1) Hub:

A hub is a common connection point for devices in a network and contains multiple ports. When a packet arrives at one port, it is copied to the other ports so that all segments of the LAN can see all packets.

2) Switch:

A switch manages the flow of data across a network by transmitting a received network packet only to one or more devices for which the packet is intended. Each network device connected to a switch can be identified by its network address, allowing the switch to regulate the flow of traffic.

3) Load Balancer:

In this paper, the load balancer implements the round robin algorithm. The Round-Robin algorithm is one of the simplest methods for distributing client requests over a group of servers. The round-robin load balancer forwards a client request to the first server in the list of servers, and then to the second server, and so on all the way to the last server in the server list. When it reaches the end of the list, the load balancer returns to the first server in the list, and the process continues. The main benefit of round-robin load balancer is that it is extremely simple to implement.

7. SDN Implementation

Using Mininet, the datacenter network topology is implemented and the configuration which is pushed on the virtual switch by the controller became simple and independent of the vendor and operating system. The controller is developed using Python and implemented on Ubuntu Operating System. Using the Python program running on the controller, the configuration could be changed to match any vendor.

Information was needed to evaluate the time required to implement a hardware-based hub, switch and load balancer in addition to software-based hub, switch and load balancer. The time required to deploy an orthodox hardware load balancer varies depending on a number of factors such as the efficiency of the engineer, the availability of the site and the urgency of the requirement. The methodology utilized in this work was completely based on experience of professional graduate students instead of industry experts.

After collecting the data using the above research methodology, the software-based hub, switch and load balancer were implemented using the following strategies:

7.1. Emulation

First, the virtual network topology was implemented using the GUI version of Mininet (Miniedit). The test topology was configured to have a total of five hosts: three servers and two clients connected to the Open vSwitch. The hosts were made to start with a dynamically assigned but easily readable IP and MAC addresses. The Open vSwitch was configured to connect to the remote POX controller on a TCP socket and communicate using the OpenFlow protocol. When the POX controller is started it opens a GUI interface that was developed using C# to manage the hub, switch and load balancer scenarios. The controller thereby implements their functionality in the SDN.

Second, the three servers were configured as Web Servers and port 80 was opened to listen to HTTP GET Requests. Meanwhile, the remaining two hosts were configured as clients which generate traffic to the servers. The three scenarios work as follows:

1) *Hub*: the Open vSwitch listens to all incoming packets and broadcasts them to all hosts.

2) *Switch*: the Open vSwitch listens to all incoming packets and forwards them to their given destination.

3) *Load Balancer*: the Open vSwitch listens to all incoming packets and selects a server to create a new session. The server is selected in a round-robin fashion for every session. The algorithm uses the OpenFlow libraries to listen and forward the packets while POX controller libraries are used to read and modify packets.

7.2. External Devices

In addition to implementing the SDN in an emulation environment, the network was connected to external devices and the connectivity of the implementation was tested. Three real servers are connected to the emulator through external interfaces, and the connectivity of HTTP and FTP services were verified using Wireshark.

8. SDN Testing and Verification

This section demonstrates the testing and verification of the SDN network. It shows the different scenarios implemented using the Mininet emulator and external devices.

8.1. Emulation Results

In the following we demonstrate the results of the hub, switch and load balancer scenarios, obtained from the Mininet environment.

1) The Hub Scenario

Error! Reference source not found. shows the results obtained from Wireshark packet analyzer, and as shown, the request from h4 to server 1 is broadcasted to all nodes in the datacenter.

2) The Switch Scenario

Fig.6 shows the results obtained from Wireshark under the switch scenario. As can be seen, the packets are exchanged between the source h4 and the destination server 2 only.

3) Load Balancer Scenario

Under the load balancer scenario, the operator of the network will pass a list of IP servers. A debug message that notifies if all servers are up or if there is a problem is also sent. When h4 requests the service in the load balancer scenario, the traffic is randomly directed to one of the servers. If there is another request from the same client or another client, the traffic is directed to the next server in the list using round-robin fashion for every session. This is shown in Fig.7.

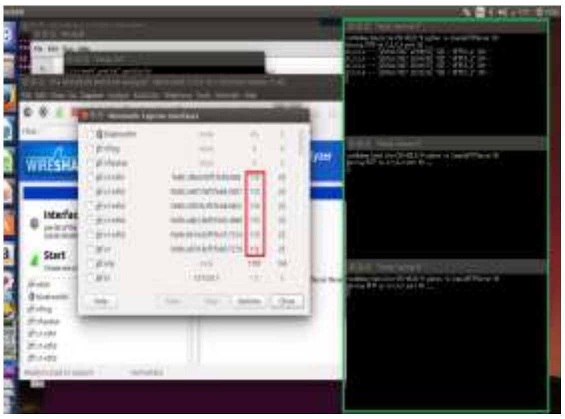


Fig.5. Broadcasting of Packets in the Hub Scenario using Emulator

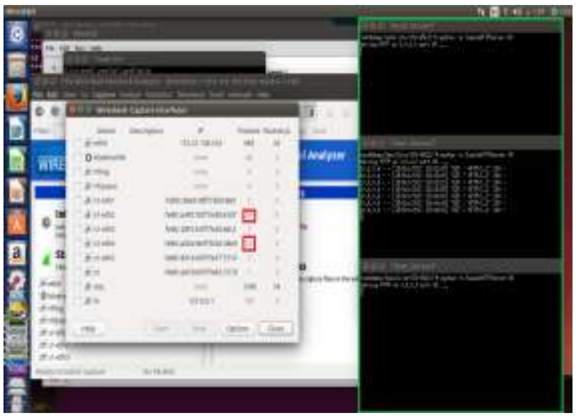


Fig.6. Exchange of Packets in the Switch Scenario using Emulator

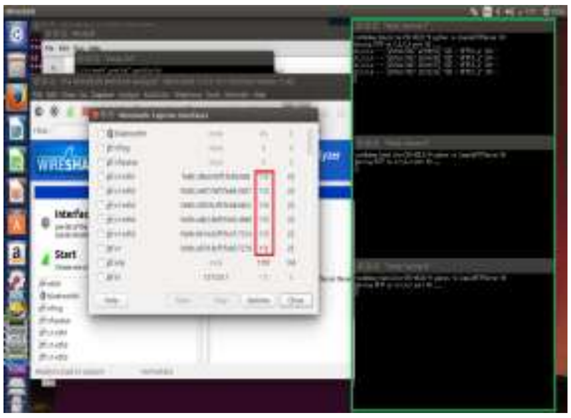
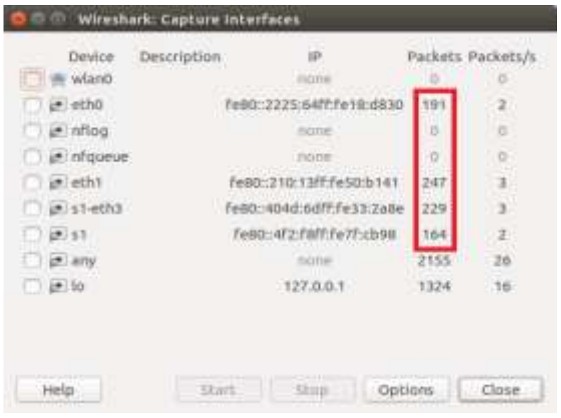


Fig.7. Traffic Distribution Among Servers using Emulator


Wireshark: Capture Interfaces

Device	Description	IP	Packets	Packets/s
wlan0		none	0	0
eth0		fe80::2225:64ff:fe18:d830	191	2
nflog		none	0	0
nfqueue		none	0	0
eth1		fe80::210:13ff:fe50:b141	247	3
s1-eth3		fe80::404d:6dff:fe33:2a8e	229	3
s1		fe80::4f2:f8ff:fe7f:cb98	164	2
any		none	2155	26
lo		127.0.0.1	1324	16

Fig.8. Broadcasting of Packets in the Hub Scenario using External Devices

8.2. External Devices Results

Here, the results obtained by testing the Hub, Switch, HTTP, FTP and load balancer connectivity using external devices are demonstrated. The three IP addresses for the three servers are (10.0.0.10, 10.0.0.20 and 10.0.0.30), and for the two clients (10.0.0.5 and 10.0.0.6).

1) Hub Connectivity

To verify the connectivity of the topology using a Programmable Hub, successful Internal Control Message Protocol (ICMP) echo and echo reply messages (ICMP ping) are sent between the nodes. To verify that the hub mechanism is working, it can be seen in Wireshark that there is a broadcast of ICMP packets inside the topology as shown in Fig.8.

2) Switch Connectivity

Fig.9 verifies the connectivity of the topology using a Programmable Switch. Successful ICMP echo and echo reply messages (ICMP ping) are sent between the nodes. Only the source and destination of the ICMP request exchange packets between them. This validates the success of the switch mechanism.

3) HTTP Connectivity

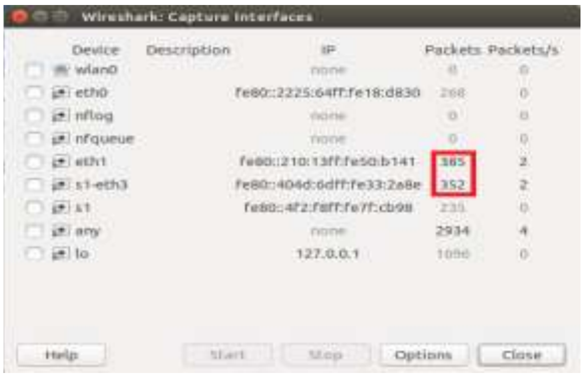
A simple test is made from the physical host to the server by opening a web browser in the host and typing http://www.sustech.com. The HTTP protocol is also verified from Wireshark as Fig.10 shows.

4) FTP Connectivity

The File Transfer Protocol (FTP) can also be verified from the web browser of the physical host or Wireshark which provides a better resolution that shows the source IP and the destination IP as can be seen in Fig.11

5) Load Balancer

In the load balancer scenario, the received traffic can be seen as depicted in Fig.12. In this scenario client 10.0.0.6 requested HTTP service. The response in the first time came from server 10.0.0.20 and in the second time from server 10.0.0.10, consistent with the load balancing mechanism.



	Device	Description	IP	Packets	Packets/s
☐	wlan0		none	0	0
☐	eth0		fe80::2225:64ff:fe18:d830	268	0
☐	nflog		none	0	0
☐	nfqueue		none	0	0
☐	eth1		fe80::210:13ff:fe50:b141	365	2
☐	s1-eth3		fe80::404d:6dff:fe33:2a8e	352	2
☐	s1		fe80::4f2:f8ff:fe7f:cb98	235	0
☐	any		none	2934	4
☐	lo		127.0.0.1	1056	0

Fig.9. Source and Destination Exchange Packets in the Switch Scenario using External Devices

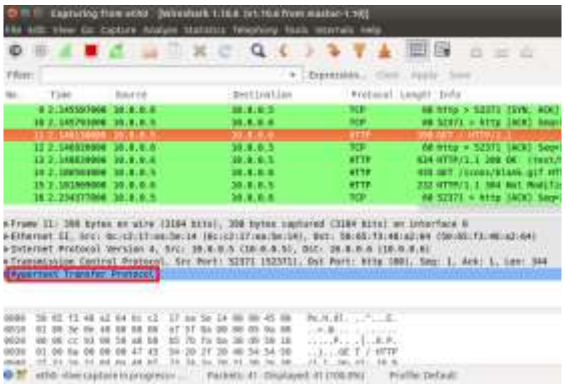


Fig.10. Exchanging of HTTP Packets using External Devices

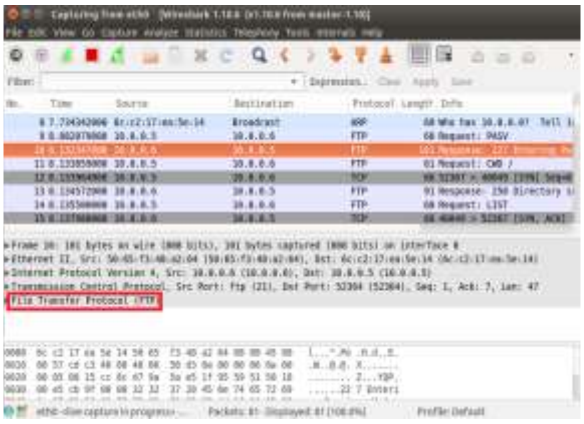


Fig.11. Exchanging of FTP Packets

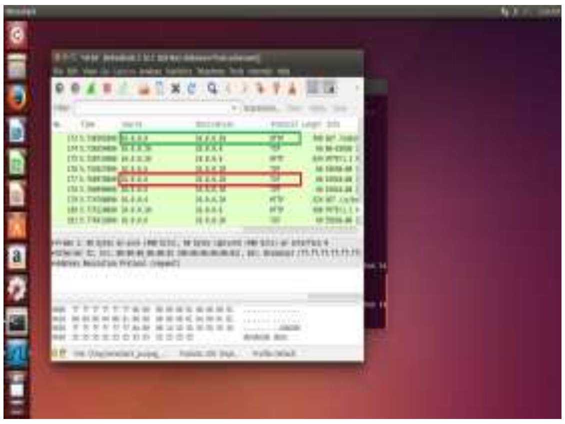


Fig.12. Distributing of Traffic Among Servers

9. Conclusions

This paper has explored and implemented the SDN approach on a datacenter network using both emulation and external devices. Three common scenarios of datacenter server options were successfully implemented: Hub, Switch and Load Balancer, where the software-based nature helped reduce the cost of implementation.

The work is vendor independent because it relies on open standards which provide flexibility in configuration and deployment by allowing the company to install the software on any white-box or OpenFlow supported device. Thus, significantly reducing the time required to deploy new services when compared to a traditional network.

For future work we plan employing other controllers other than POX to identify the controller type that improves the performance of the datacenter network under different topologies and traffic. Another possible improvement is to add a redundant controller to the network to achieve lossless, low delay performance and higher availability. In addition, the controller can be modified to implement different load balancing algorithms including Weighted Round Robin and IP-based Hashing and evaluate their performance under different traffic conditions.

10. References:

(Feamster et al., 2013) Feamster, N., Rexford, J., and Zegura, E. (2013). "The Road to SDN: An Intellectual History of Programmable Networks", *ACM Queue*.

(Mahjoub, 2015) Mahjoub, R., (2015) “Event-Driven Network Control Using Software-Defined Networking”, University Of Khartoum.

(Körner , 2015) Körner, M., (2015) “Software Defined Networking based Data-Center Services”, University of BerlinzurErlangung.

(Salih et al., 2014) Salih, H., Ahmed, W., and Ahmed, Y., (2014) “Implementation of Remote Configuration Using SDN Approach”, Sudan University of Science and Technology.

(Ghaarinejad, 2015) Ghaarinejad, A., (2015) “Comparing a Commercial and an SDN-Based Load Balancer in a Campus Network”, Arizona State University.

(Chakravarthy & Amutha, 2019) Chakravarthy, V. and Amutha, B., (2019) “Path based Load Balancing for Data Center Networks using SDN”, *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 9, Issue. 4, pp. 3279~3285.

(Aguilar & Batista, 2019) Aguilar, L. and Batista, D., (2019) “Effectiveness of Implementing Load Balancing via SDN”, *XXXVII Brazilian Symposium on Computer Networks and Distributed Systems*, Brazil.

(Cui et al., 2018) Cui, J., Qinghe, L., Zhong, H., Tian, M. and Liu, L., (2018) “A Load-Balancing Mechanism for Distributed SDN Control Plane using Response Time”, *IEEE Transactions on Network and Service Management*, vol. 15, Issue 4, pp. 1197-1206.

(Sufiev et al., 2019) Sufiev, H., Haddad, Y., Barenboim, L., and Soler, J., (2019) "Dynamic SDN Controller Load Balancing", *Future Internet*, Vol. 11, Issue 3, pp. 75.

(Sitota, 2021) Sitota, K., (2021) "Software Defined Network-based Dynamic Load Balancing Algorithm using Floodlight in Hierarchical Networks", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 10, Issue.

(Mulya et al., 2019) Mulya, F., Purboyo, T., and Latuconsina, R., (2019) "Experimental Study of Load Balancing on Software Defined Network using Ant-Colony Optimization", *Asian Research Publishing Network (ARPN) Journal of Engineering and Applied Science*, Vol. 14, Issue 17.

ONF (2012). *Software-Defined Networking: The New Norm for Networks*. Open Networking Foundation.

(Xia et al., 2015) Xia, W., Wen, Y., Foh, C., Niyato, D. and Xie, H., (2015) "A Survey on Software-Defined Networking", *IEEE Communication Surveys & Tutorial*, Vol. 17, Issue. 1.

(Ghosh & Manoranjitham, 2018) Ghosh, A. and Manoranjitham, T., (2018) "A Study on Load Balancing Techniques in SDN", *International Journal of Engineering & Technology*, Vol. 7, Issue 2, pp. 174-177.

Load Balancing Algorithms (2021, May 22) Kemp Technologies https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques/

Mininet Overview (2019, August 17) Mininet, http://mininet.org/overview/

Doi: doi.org/10.52132/Ajrsp.en.2023.45.2