Software Defined Networks: A case for QoS implementation at the Greek School Network

John Christodoulopoulos[1], *Michael Paraskevas[1,2], Vasilis Triantafyllou[1]

[1] Computer Informatics and Engineering Dept., Technological Educational Institute of Western Greece, Greece
[2] Computer Technology Institute and Press "Diophantus", Patras, Greece
mparask@cti.gr

ABSTRACT

The Software Defined Networking is a different approach to the way network equipment operation. Until recently, network devices embody both the physical layer (signaling) and the logic on how to exchange packets. This was intended to maximize the upload speed, and a predetermined path are not imposed delays. Similar solutions have been given to other resources in the field of Information Technology. Processing power, magnetic storage media, as well as random access memory, a factor which may be available depending on the needs of users. This was the main idea of the area of mainframes up to Virtualization and Cloud Computing. SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow protocol is a foundational element for building SDN solutions. In this paper an theoretical study of SDN technology will be given, as also as an implementation of a QoS mechanism based on SDN technology will be presented for the Greek School Network.

Keywords

Software-Defined Networking (SDN), OpenFlow, QoS, Greek School Network.

1. INTRODUCTION

As the complexity of networks is increasing and the equipment becomes increasingly expensive, cases of underutilization of network resources strengthened the search for a different way of using the natural level of communication [1]. A way that would allow the sharing of material from different user groups and the flexible allocation of additional resources to those groups with higher data exchange requirements. Solving the above problem would come only by separating the decision level in relation to the physical signaling level. Furthermore, flexibility means easy change of equipment behavior, a behavior-based metrics use of communication channels in order to decide the required data traffic rate. This effort is now termed as Software Defined Networking (SDN) [2].

DOI: 10.1145/3003733.3003768

SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow™ protocol [3] is a foundational element for building SDN solutions.

So following these examples, the software defined networking provides data streams according to existing needs and available equipment on which it operates. It is an abstraction mechanism allows the handling of communication channels by the software and not by the material itself.

According to the official definition of the Open Networking Foundation "The Software Defined Networking (SDN) is an emerging network architecture, wherein the control system is separated from the forwarding system and is directly programmable. This migration control, formerly connected to individual network devices (routers, switches) is accessible to computing devices allowing the underlying infrastructure to pump applications and network services, which can cope the network as a logical or virtual entity" [1]. Another definition of SDN is the following: "The physical separation of network control plane from the forwarding plane, and wherein the control level controls several devices". [4]

This work is organized as follows: Section 1 introduces Software-Defined Networking. Section 2 outlines OpenFlow™. Section 3 describes our QoS implementation at Greek School Network. Finally, conclusions and future work are presented in Section 4.

1.1 The SDN technology

Distributed control and network transport protocols run in routers and switches, allowing the network packets to travel inside the web world [2]. Despite their widespread acceptance the traditional IP networks are complex and difficult to manage. In order to achieve an efficient network the network operators policy should regulate separately and with special commands on each network device. This leads to a complex configuration, thus making the network vulnerable to errors and the need to restart the device to upload changes to the settings. So the current IP networks do not allow their immediate configuration changes and application of network devices.

The control plane, which is responsible for the management of network traffic and the data layer, which promotes the traffic according to the control level decisions are embedded in network devices, thus the network infrastructure is not flexible to change and may not be able easily to evolve.

The Software-Defined Networking technology (SDN) offers changing limitations of current network infrastructure, ie the

separation of network control logic of the device of routers and switches, which promote the traffic of packets. This separation makes simple forwarding devices, the control logic is implemented in a central controller, resulting in easy implementation of policies, configuration and network evolution. This separation is carried out by a programmable interface (interface) between the switches and the SDN controller. The controller allows to control the status of the components of the data layer through an Application Programmable Interface (API), e.g. of OpenFlow.

The technology of SDN and OpenFlow originally started experimentally from academia, but quickly backed by large companies such as Google, Facebook, Yahoo, Microsoft, Verizon and Deutsche Telekom, thus founding the Open Networking Foundation (ONF) with the single purpose of the development, promotion and acceptance of SDN.

As defined the SDN refers to a network architecture where the forwarding traffic in data level is manageable from a remote control plane, separated from the devices, Figure 1.

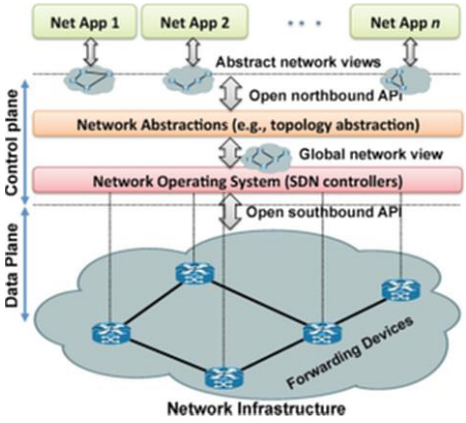


Figure 1. SDN Architecture

The architecture of SDN is comprised of four pillars:

1. The control plane is separated from the data plane.
2. The promotion decisions are based on flows and not on the basis of their destination.
3. The control logic is an external entity called SDN Controller or Network Operating System (NOS).
4. The network is programmable through applications running on the Controller.

In terms of network architecture, the SDN is expected to change the way engineers and network designers build and activate their networks. With the introduction of the SDN, the networks have been open standards, non-specialized hardware - software and easy programming and management. The SDN will give businesses and suppliers more control over their networks, allowing adjustment and optimization to reduce the overall cost of their networks.

Some of the main benefits of SDN is [5]:

- *Simplicity in network management*: With the SDN network can be viewed and configured as a single node that takes into abstract level demanding network management tasks to be easily managed through interfaces (interfaces).
- *Rapid development services*: The new network features and applications can be developed so quickly within a few hours instead of several days.
- *Automation settings*: manual configuration tasks, such as assigning VLAN and QoS setting can be done automatically.
- *Network Virtualization*: Once the virtualized servers and storage devices developed before the networks can benefit and virtualization of SDN.
- *Reduction of operating costs*: With the benefit of automation of network development, one change in it has never been more easy process and result in reduced costs of network operation.

1.1.1 Historical development of SDN technology

In 1980 there was the sense of the centralist control network as data and control were made through the same channel, there were specific frequencies for initialization of the line or to route the telephone call. Later, in 1981 AT & T separate Data level from Control level via an entity called Network Point which could easily understand if the line was engaged.

In 1990 the Active Networks emerged as a concept of programmable networks, which consisted of switches where made some calculations on packages, such as trace in each router, firewalls, proxies and some applications. Research of DARPA 1994-95 [6] finds problems in these networks, such as difficulty adapting to new technologies, low performance due to the multi-level protocols and difficulties in new services. It was something innovative and took 10 years to become a model, to obtain the market, implemented and developed. It consists of active nodes that enable routers to give the ability to the infrastructure of the new network services that makes it look to the Software Defined Networks (SDN's). It has two approaches, the Capsules, where each message is program and evaluated by the Node carrying packages and Programmable Switches which functions run on routers and packets are routed through programmable nodes as the program runs for routing of the packet depends the header content of the packet. This technology has some disadvantages, such as the concept of time was not clean, had expensive devices, there were errors in security, since the packets containing the program code and network interoperability, i.e. the ability to send and receive constantly packages for the network's QoS without causing the network. All these problems have led to the conclusion that there had to be an embodiment where it would be compatible with the backward technology and operate on the same infrastructure. This solution gives us the OpenFlow protocol.

Also, in 1990 has developed the idea of Network Virtualization, which gives us the representation of one or more network topologies on the same network infrastructure. As such cases we find the VLAN, Tempest, Vini, Cabo, VMWare and Nicira [7]. They give us many advantages, such as sharing of resources as many routers work on one platform and use the CPU, RAM, Forwarding Tables and Bandwidth as well as a possibility of personalization in which we can have our own software for routing policies and forwarding through general purpose CPU and Network processors & fpga's for Data Level.

All these elements are a legacy for SDN why:

- Separate the services of the network infrastructure.
- We can have multiple Controllers to a Switch.
- We can have multiple logical topologies.

But to have the control in a packet switched network will need to have the separation of control plane from the Data level.

The control plane is the logic for controlling the forwarding behavior of packages, e.g. routing protocols, filters, firewalls, NAT, security, load balancer.

The Data Plane is responsible for promoting the movement in accordance with the logic of the control level, e.g. ip forwarding - Layer 2 Switching.

The separation of these levels leads us to some advantages:

- *Faster innovation*: The logic of control is not tied to the hardware and so we have the independent development and growth of the Controller of Network software in connection with the network infrastructure.
- *Wide display network*: We can easily infer the behavior of the network, thus allowing control of the network by a high-level program we see the network debugging and tick its functions.
- *Flexibility*: Presentation of new services more easily.

The development of control in packet switching networks began in 2003. There the FORCES - [8] uses protocols for multiple Control Elements and Forwarding Elements gives an idea like OpenFlow but needed to become standard , to adopt and create new material.

In 2004 there was the idea of using existing protocols for calculating routes to packet forwarding, called Routing Control Platform [8]. Here, the calculation was made in a point of the Autonomous System and when it find's the routes advertised via iBGP protocol, but control is limited by the support of existing protocols.

In 2007 has created the Ethane [8] and works as a network architecture for operators and is based on direct Performed network policies. The Domain Controller calculates the flow tables based on policies that are needed but modified switches, as OpenWrt-NetFPGA and Linux to support the Ethane.

In 2008 we have the presentation of OpenFlow™. The controller speaks with the forwarding tables of the switch and that is needed is to open manufacturers an interface to receive the switch, the traffic flows from the controller.

1.2 SDN Architecture

A SDN architecture can be described as a composition of different network layers, as shown in Figure 2 (b). Each layer has its own specific features, such as southbound API, NOSs, northbound API, network applications and other [2].

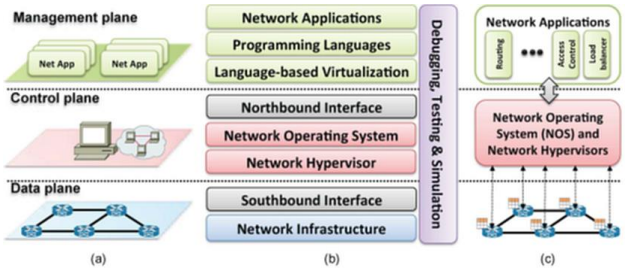


Figure 2: Software-Defined Networking architecture at levels (a), layers (b) and system design (c)

Figure 2 (a) shows the general architecture of the SDN three levels. The layers of the SDN shown in Figure 2 (b), and an illustration of the system architecture design have in Figure 2 (c).

The SDN infrastructure is almost identical to that of traditional networks, which consist of network equipment (routers, switches, etc.). The main difference lies in the fact that physical devices are simply forwarding data traffic without an embedded software to control or make forwarding decisions to the packages. The "intelligence" of the network is a central control system called Network Operating System (NOS). These networks are above of some interfaces (OpenFlow), which allow the dynamic programming of heterogeneous forwarding elements which was difficult so far.

The Northbound and Southbound interfaces are two basic abstractions of SDN system allowing the communication with the upper and lower network levels, respectively.

Network applications can be considered as the "brains of the network", which implement the control logic to be translated into commands to be installed in the data level, defining the behavior of forwarding devices.

2. THE OPENFLOW TECHNOLOGY

The OpenFlow™ is an approach that enable network administrators to implement and control the data they want in the software, without waiting the manufacturers to integrate a new version of firmware. Furthermore, enable manufacturers to give access to researchers in terms of their equipment in a unified way without opening up their products. Therefore, researchers are able to carry out experiments with new protocols in a real network without affecting the traffic of the production network.

The OpenFlow™ use flow tables which is similar to lookup tables, in modern Ethernet routers and switches. These flow-tables can implement firewalls, NAT, QoS or to collect statistics for network management without interfering with other manufacturers. Also, the flow-tables contain the match / action rules that can be created and modified by a centralized controller. The controller offers a programming control of flows to the network administrator to specify a path from source to destination using processing based on the flow of packet forwarding. Reduces power consumption and management costs by eliminating the processing of packets in the router, since most routes of packets specified by the centralized controller.

An OpenFlow switch consists of a flow-table, which performs search and forwarding procedures in the packages, and a secure channel to an external controller, as shown in Figure 3. The controller manages the switch through the secure channel using the OpenFlow protocol [3].

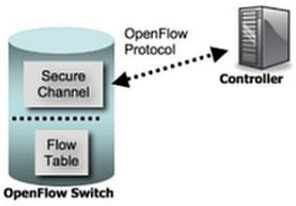


Figure 3: OpenFlow enabled switch OF v1.0 specification

2.1 OpenFlow Analysis

2.1.1 Architecture

The OpenFlow network architecture consists of three basic concepts:

- OpenFlow-compatible switches that make up the data level.
- The control plane consists of one or more OpenFlow controllers.
- A secure control channel connecting the switches to the control level.

The switches communicate with each other and with end users (hosts) using the data path software (data path) that is provided and the controller communicates with switches using the control path as shown below, Figure 4.

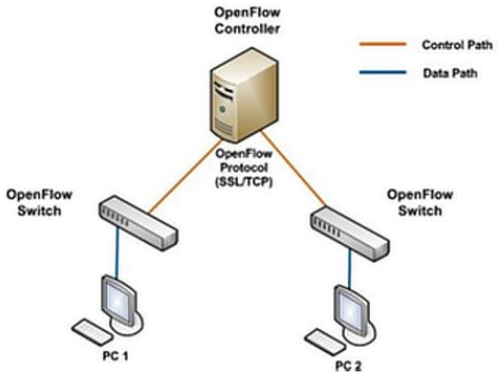


Figure 4: The network architecture of OpenFlow

2.1.2 Communications

There are three categories of communication in OpenFlow protocol:

- Controller-to-Switch: is responsible for detecting characteristics, configuration, programming switch and retrieve information (message types: Features, Configuration, Modify-State, Read-State, Send-Packet, Barrier).
- Asynchronous: communication initiative of OpenFlow compatible switch without any request from the controller. It is used to inform the controller of packet arrivals, if the switch status changes and errors (message types: Packet-in, Flow-Removed, Port-status, Error).
- Symmetric: communication to see if the control channel is active and available (message types: Hello, Echo, Vendor).

2.1.3 Forwarding Mechanism of Packages

In an OpenFlow network when the switch receives a packet, analyzes the header and check if they fit in the flow-table rules field. If there is a match, then the action carried by the flow-table. If packages fit more than one rules, then the packets are compared with a specific record-flow based on hierarchical priorities, namely selected record-flow with the highest priority. Then, the switch updates the counters of that table-flow. Finally, the switch forwards the packet to an exit port. If the incoming packet does not match any entry-flow in flow table, the switch will forward the packet to the controller to calculate the logic to be implemented in the package and similar future packets. The process mechanism of packet forwarding illustrated in the flow chart, Figure 5.

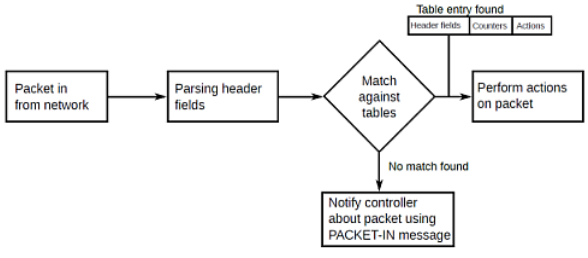


Figure 5: Forwarding Mechanism of OpenFlow™

2.1.4 Versions of OpenFlow

Apart from the version 1.0 of OpenFlow as described in the preceding paragraphs there are other versions that will be briefly described in connection with the differences in version 1.0 [3].

The OpenFlow 1.1 was released in February 2011 [9]. It contains significant changes compared to the OpenFlow 1.0. In version 1.1 the two main changes are a pipiline consisting of multiple flow-tables and a group-table.

The OpenFlow 1.2 was released in December 2011 [10]. It comes with extensive support of the Protocol in relation to IPv6. The OpenFlow 1.2 can match the IPv6 source and destination addresses, protocol number, flow label, traffic class in the various fields of ICMPv6. Also, one switch can connect to more than one controller.

The OpenFlow 1.3 [11], introduces new capabilities for Monitoring Operations, Management (OAM), through a panel-meter (Meter-table in the switch architecture). The meter is directly connected to a recording-flow table from the meter ID and measures the percentage of packets that have been assigned. So simple or complex frames QoS can be implemented with the OpenFlow 1.3 and subsequent versions.

The OpenFlow 1.4 was released in October 2013 [12]. The ONF improved support for OpenFlow Extensible Match (OXM). Added TLV (type-length-value) structures for ports, tables and queues in the protocol, and hard-coded portions of the previous editions have been replaced by new TLV structures. The setting of the optical ports is now possible. In addition, controllers can send control messages to a single set of messages to switches. Including also small improvements in the group boards and monitoring capabilities.

2.1.5 OpenFlow Controllers

The controller is the core and the main part of the network operating system (NOS) in SDN. It is responsible for handling the flow-tables of the switch and for communication between applications and network devices using the OpenFlow protocol.

The controllers can be classified into two main categories [13]:

1. Open source, individual controller for research and development.
2. Commercial, closed source, distributed controllers.

3. IMPLEMENTING QoS

3.1 Introduction to the Scenario

To simulate real network will need some tools that will enable us to implement the network. For this reason will be the use of Virtualbox, which is an environment that allows us to create virtual machines and describe the simulation experiment. Also, in the case of the logical switch will choose the OpenVSwitch (OVS). Also, we use the Floodlight controller as logical switch controller. All tools work fine in an Ubuntu-based linux OS in a Laptop machine.

As part of the Greek School Network will implement a virtualization environment in the case of school laboratories. These are generally in the same network as the office of the Director and teachers and share the same ADSL link to connect with the Internet.

A use case in this network would be what would happen if the laboratories made excessive use of the internet line and eg the Director's office cannot have a quality teleconference. The answer to this scenario is create traffic queues in the switch (OpenVSwich) and in this way will reduce the traffic on the communication channel, allowing flow-traffic through the controller to drive the data packets in the traffic queues thereby reduce bandwidth in the communication channel.

3.2 Experiment

3.2.1 Starting Floodlight Controller

First step is to start the Floodlight as OpenFlow controller for the OVS with the command:

- sudo java –jar target/floodlight.jar

3.2.2 Setting Up OpenVSwitch

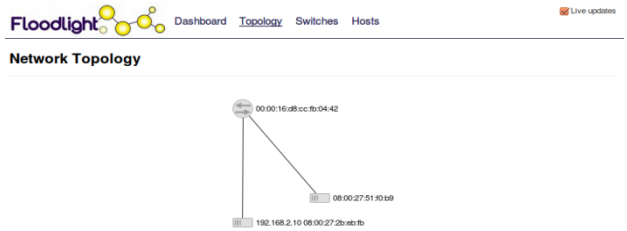
Second step is to setup the OVS and attach to the floodlight. In additional create two virtual interfaces for the hosts.

- ovs-vsctl add-br sdnbr
- ovs-vsctl add-port sdnbr wlan0
- ifconfig wlan0 0 up
- ifconfig sdnbr 192.168.2.3 netmask 255.255.255.0 up
- ovs-vsctl set-fail-mode sdnbr secure
- ovs-vsctl set-controller sdnbr tcp:192.168.2.3:6653
- ip tuntap add mode tap vnet1
- ip link set vnet1 up
- ovs-vsctl add-port sdnbr vnet1
- ifconfig sdnbr up
- ip tuntap add mode tap vnet2
- ip link set vnet2 up
- ovs-vsctl add-port sdnbr vnet2
- ifconfig sdnbr up
- ip link
- ovs-vsctl show

And after this commands we will start Virtualbox and attach the virtual machines to the virtual interfaces that we have created (vnet1 & vnet2).

3.2.3 Build Network Topology

Our network topology consist from two hosts and one switch. As shown below.



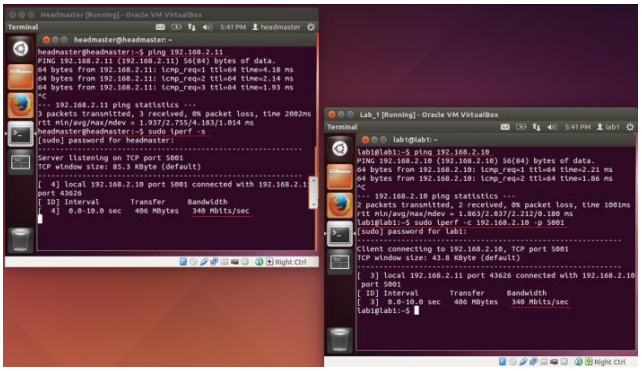
The first host (VM-Headmaster) represent the Director's office computer and the second host (VM-Lab1) represent the laboratories connecting by the switch which is the logical software switch OpenVSwitch. So to build our network topology we start the two VM's in Virtualbox and attach the 2 virtual interfaces in OpenVSwitch to have the connection.

3.2.4 Setting Flows for traffic connection

To interconnect the two virtual machines and communicate with each other will give the OpenVSwitch two traffic flows through the controller:

- curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11","actions":"output=2"}' http://192.168.2.3:8080/wm/staticflowpusher/json
- curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10","actions":"output=3"}' http://192.168.2.3:8080/wm/staticflowpusher/json

To verify the communication we can execute ping command between the two virtual machines (VM) and to check the existing bandwidth between the VMs we use the iperf command.



At this step the bandwidth between two host is very high about 340 Mbits/sec and we start the QoS Scenario by setting up OVS.

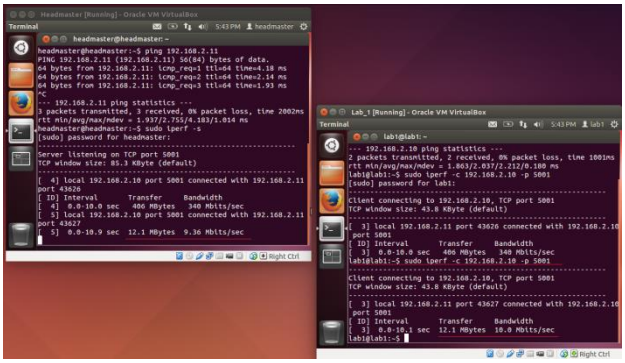
3.2.5 Setting Up QoS on OVS

At this point we create 2 traffic queues for delivering the packets to limit the bandwidth between two hosts. One queue is the fast speed about 10 Mbits/sec and the second queue is the slow speed about 2 Mbits/sec.

So for the QoS on OVS create the queues to each virtual interface:

- ovs-vsctl set port vnet1 qos=@newqos -- --id=@newqos create qos type=linux-htb queue=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@q1 create queue other-config:min-rate=2000000 other-config:max-rate=2000000
- ovs-vsctl set port vnet2 qos=@newqos -- --id=@newqos create qos type=linux-htb queue=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@q1 create queue other-config:min-rate=2000000 other-config:max-rate=2000000

and verify the fast speed and set the bandwidth about 10 Mbits/sec.



3.2.6 Setting Up QoS flows with enqueue traffic

To make use of the second traffic queue (slow speed) will empty the table of flow switch and added to the flows with exit the second queue for implementing the QoS

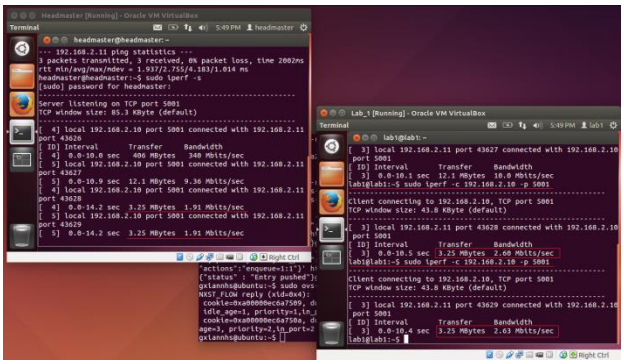
First we delete the existing flows with the command:

- ovs-ofctl del-flows sdnbr

and then to interconnect the two virtual machines and communicate with each other with the slow speed will give the OpenVSwitch two traffic flows through the controller:

- curl –s –d '{"switch":" 00:00:16:d8:cc:fb:04:42","name":"flow-1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11","actions":"enqueue=2:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json
- curl –s –d '{"switch":" 00:00:16:d8:cc:fb:04:42","name":"flow-2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10","actions":" enqueue =1:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json

and verify the slow speed and set the bandwidth about 2 Mbits/sec.



The experiment of QoS implemented using the SDN and we see successfully reducing the bandwidth of the network.

4. CONCLUSIONS

The present study describes in detail the theory for the technologies of Software-Defined Networking and OpenFlow™ of presenting the details separately for each technology. We simulate a real network in a virtualization environment (virtualization), which refers to the topology of a School Network by taking information from the Greek School Network. We approached successfully implementing such a network using the SDN and considering the provision of quality of service (QoS) to the end users, we implemented after research analysis one network policy that offers QoS. The importance of a quality of service (QoS) in a network environment is quite important and useful. For this reason, the considered as a use case of the implementation of SDN why can effectively solve QoS problems that exist in a large number of networks.

However SDN offers many benefits solving these major problems and implements QoS cases easily from a central control level because it gives real interaction setting in each node of the network, such as routers, switches, servers, load balancing, firewalls, etc. Of course, the SDN is not the only solution to all the problems in the network area. It will certainly help in need fewer settings through the console (CLI) of network nodes, but the support of organizations and their transition to SDN models will help in its development as more functions will be discovered.

5. ACKNOWLEDGMENTS

The related activities that led to these results were performed in the context of the cooperation between the Greek School Network Directorate of the Computer Technology Institute and Press "Diafantus" and the postgraduate MSc Course "Technologies and Infrastructures for Broadband Applications and Services" (http://www.msc.cied.teiwest.gr/mscen/), of the Computer & Informatics Engineering Department – CIED (Technical Educational Institute of Western Greece).

6. REFERENCES

[1] Open Networking Foundation, "Software-defined networking: The new norm," ONF White Paper, Apr. 2012.

[2] D.Kreutz, F.M.V.Ramos, P.E.Verissimo, C.E.Rothenberg, S.Azodolmolky, S.Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, Contributed Papers, 2015.

[3] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," 2014.

[4] Open Networking Foundation, "Software-Defined Networking (SDN) Definition," 25 May 2014. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition.

[5] M.Nispel, "A Different Take on Software-Defined Networks," Ebook, Extreme Networks, 2014.

[6] N. Feamster, M1.2 SDN History: Programmable Networks, 2014. [Online]. Available: https://www.youtube.com/watch?v=CFNTzni1FhA&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=3.

[7] N. Feamster, M1.3 SDN History: Network Virtualization, 2014. [Online]. Available: https://www.youtube.com/watch?v=vsbyyNFg5BM&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=4.

[8] N. Feamster, M1.4: SDN History: Control of Packet-Switched Networks, 2014. [Online]. Available: https://www.youtube.com/watch?v=AFj-ZIuAGwo&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=5.

[9] B. Heller, "OpenFlow Switch Specifications", v.1.1.0, 2011.

[10] Open Networking Foundation, OpenFlow Switch Specifications version 1.2.0, 2011.

[11] Open Networking Foundation, OpenFlow Switch Specifications version 1.3.0, 2013.

[12] Open Networking Foundation, OpenFlow Switch Specifications version 1.4.0, 2014.

[13] A. Sonba and H. Abdalkreim, "*Performance Comparison Of the state of the art Openflow Controllers", MSc Thesis at* Master's Programme in Computer Network Engineering, University of Halmstad, 2014.