# Enhancing the Performance of OpenFlow Network by Using QoS

Ammar D. Jasim and Dalia A. Hamid

**Abstract**— OpenFlow (OF) is a new technology started in 2008 and managed by the non- profit organization called "Open Network Foundation", and it is one of the standers of Software Defined Networking (SDN). SDN approach differs from traditional network in the way that the control plane is separated from the data plane, the data plane is remains in the networking device (switch) while the control plane is removed to a separate device called the controller and it is the responsible for controlling the packets in the data plane, The control and data planes communicate with each other by OF protocol. While in traditional network, the networking device contains both the data and control planes, and the switch is responsible for controlling the packets. This separation in SDN allows the manager and researchers to run their own code for controlling and managing the networking devices.

This paper focusing on study and evaluate the performance of four types of SDN controllers (Floodlight, Beacon, Open-MUL and Open-IRIS) in OpenFlow network that built in Mininet emulator and then enhance the performance of the network by using QoS technique with Floodlight controller. This evaluation has been done by using three types of traffics: ICMP, TCP and UDP.

Form the results of the evaluation, it can be seen that the SDN controllers show different behaviours that return to the reason that each controller uses algorithm to distribute the packets between the threads and also the mechanisms and libraries that have been used by each OF controller. And also the performance of the network have shown better results when QoS have been used in the OF network.

**Index Terms**— OpenFlow (OF), Software Defined Networking (SDN), QoS, ICMP, TCP, UDP, Mininet.

## 1 Introduction

In 2008, McKeown group at Stanford University presented a new protocol named "OpenFlow" to the network community [1]. OpenFlow (OF) is the first standard protocol that was designed specifically for Software Defined Networking (SDN). Its essential use case is to allow the removal of the native control plane from the network device to a remote central device. OF is now owned by the Open Networking Foundation (ONF) [2].

In communication networks, the idea of having a centralized control plane was far-fetched despite the attempts which had been ignored by the network community. However soon OF did it.

The first main reason for that development is for cloud servers. The providers of cloud service were facing big challenges because of the increasing in cloud services (increasing the operations in data centers). The networks were designed and run in the last thirty years with no enough flexibility to tolerate an operation of new required services [3].
The server virtualization and cloud computing have changed the way of using the data centers. Virtualization allows more efficient use of IT resources and greater levels of control. The cloud computing extends these benefits, by allowing the organizations to reduce the complexity of their infrastructure, reduce the load of work on staff, and have more rapidly scalable models. Both of these technologies enable the organizations to better meet their demand [4].

But despite of the rapid development of virtualization and cloud computing, most of the current network technologies have not been developed to consider virtualization and cloud computing. So the static topologies still require manual intervention to deploy and migrate virtual machines, that increase the cost and the network may have a bottleneck in the future and all that affect the organization's ability to respond quickly to the changes in the environment [4].

Many companies use OF protocol within their data center networks to simplify their operations. OF and SDN allow data centers and researchers to innovate their networks with new ways because it is easier to abstract the network [5].
The second main reason for this development is that the implementation of OF is not restricted in software but can also be implemented in hardware and that allows the adopters to make experiments with it immediately, such as Google, the research team designed new networks based on SDN concept [3].

## 2 SDN Architecture and OF technology

The ONF is the group that is most associated with the development and standardization of SDN. According to the ONF, "SDN is the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices." [6].
The architecture of SDN is divided into three layers [7], as presented in Fig. 1.

a) Infrastructure layer: it is the foundation layer that represents the forwarding hardware in the SDN network architecture. When the controller needs to communicate with the network infrastructure, it requires certain protocols to control and manage the various devices of network equipment; it uses the most popular southbound (SB) protocol called OF protocol.
b) Control layer: presents an abstract view of the complete network infrastructure, enabling the adminis-

trator to apply custom policies across the network hardware.

c) Application layer: it consists of network services and applications. The northbound (NB) APIs represent the software interfaces between the application and control layers.

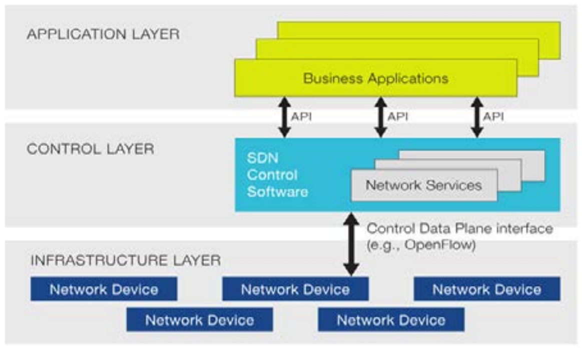


Figure 1 SDN architecture

The benefits of SDN architecture:

a) Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.
b) Agile: Abstracting control from forwarding lets the administrators dynamically adjust network-wide traffic flow to meet changing needs.
c) Centrally managed: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
d) Programmatically configured: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which can be written by them because the programs do not depend on proprietary software.
e) Open standards-based and vendor-neutral: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

In OF technology, OF is considered the first SDN standard managed by ONF that defines the protocols and functions to manage the switches via centralized controller. OF is a protocol that provides the communication between the controller and the switches over secured channels to enable the controller to manage the forwarding tables in the switches and to configure the network devices. The designing and managing of network is not the job of this protocol but it's up to the vendors and the implementers to decide that. The OF architecture consists of three components: OF switch, OF controller and OF protocol.

OF switch is the basic device for forwarding the packets according to its table that is similar to the table in the traditional network but the difference is that the table is not managed by the switch. An OF switch contains a flow table that perform the packet lookup process and forwarding, and a channel to an outer controller. The controller communicates with the switch through this channel using OF protocol as presented in Fig. 2 [8].

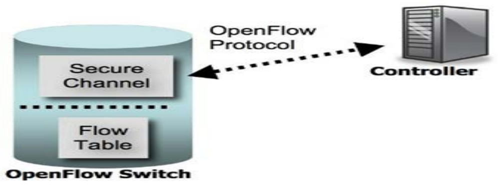
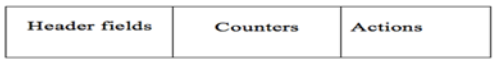


Figure 2 Connectivity between OF switch and OF controller

The flow table in each switch contains a collection of flow entries, each flow entry contains three fields: header, actions and counters as presented in table 1.

**Table 1** Flow entry in OF table

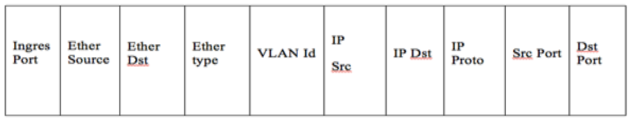


| Header fields | Counters | Actions |
|---|---|---|

Header field contains ten fields to compare the incoming packets with these fields. Table 2 presents these fields.
While the counters field are used for statistics purpose, to pursue the number of bytes and packets of each flow and also the elapsed time since the flow initiation.
And finally, the actions field defines the way to process the packets, the action could be:

1. Forwarding the packet to a specific port or ports.
2. Dropping the packet.
3. Forwarding the packet to a controller.

**Table 2** Fields to match the packets against flow entries



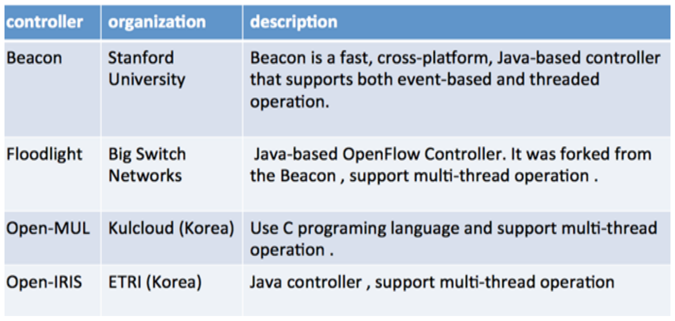
| Ingres Port | Ether Source | Ether Dst | Ether type | VLAN Id | IP Src | IP Dst | IP Proto | Src Port | Dst Port |
|---|---|---|---|---|---|---|---|---|---|

Today, many OF controllers exist and each one has its own properties. Surely there is a question of how to choose the suitable controller? The answer is based on the needs of the researcher like the supported interfaces, programming language, GUI supports, etc. Figure 3 shows the OF controllers that have been used in this work.

## 3 QoS in OpenFlow Network

Today in networks, to monitor the QoS parameters such as bandwidth, delay and packet loss is fundamental to ensure that the operations of multimedia applications are smooth. This is often done by sending traffic through the network. These measurements can only perform in non-peak hours because these measurements could affect on the production traffic.

With SDN approach, the network has the ability to perform



| controller | organization | description |
|---|---|---|
| Beacon | Stanford University | Beacon is a fast, cross-platform, Java-based controller that supports both event-based and threaded operation. |
| Floodlight | Big Switch Networks | Java-based OpenFlow Controller. It was forked from the Beacon , support multi-thread operation . |
| Open-MUL | Kulcloud (Korea) | Use C programing language and support multi-thread operation . |
| Open-IRIS | ETRI (Korea) | Java controller , support multi-thread operation |

Figure 3 OpenFlow controllers

the QoS measurements based on the production traffic without affecting on it [2].
OF community recognizes the importance of QoS support by performing the OF protocol version 1.0 with several QoS capabilities. OF switch provides a limited support to QoS through the queuing mechanism where one or more queues can be attached to one of the OF switch's port to map the flows on it. These mapped flows will be handled according to these configured queues such as the minimum and maximum rate [8].
The other mechanism is the rate limiting where ingress rate and ingress burst are specified. OF switch will drop any packets beyond these specified rates [9].
When the switch starts its connection to the controller, it should send a report to the controller of its QoS capabilities like the supported classes. A certain level of service can be assigned to each flow entry in the table of the OF switch, this service is called a service class. Then the assigned flow will place in the respective service queue. Three QoS primitives is proposed by OF organization to be inserted into the OF switches. These primitives are [5]:

1. Minimum Rate: also called slicing, that gives a minimum bandwidth to each flow at each egress.in the case that the switch is not able to give the minimum rate to the egress of the flow, then the switch permit the flow to be part of a queue that called a service class. The setting and configuring of the queues will be done by the OF controller to arrange each flow in a specific queue. The controller does this process to adjust the treatment of the packets in the switch.
2. Rate Limiting or Maximum Rate: designed for limiting the rate of the packet flow.
3. Strict Precedence: to assign a priority to the flows in the OF switch table.

## 4 Experimental Setup

The goal of this work is to evaluate the performance of a custom network built in Mininet emulator. This evaluation has been done by implement the OF controllers and three types of traffic were used during the evaluation: ICMP, TCP and UDP traffics. Then propose a method to enhance the network.
The purpose of ICMP traffic is to measure the additional delay that obtained from the controller to process the first packet of a flow. The additional delay comes from the connection between the switch and the controller. Ping tool was used to generate this traffic.
The purpose of TCP traffic is to see how the transfer time of TCP packets will be affected by the communication between the switch and the controller. By Implementing a TCP client-server application, this traffic will be generated. The measurement of transfer time has been done by using different sizes of data, the measured time include the time from the establishment of the connection until the close of the connection after sending all the bytes. Iperf tool was used to generate this traffic.
Since UDP is the protocol with no retransmission process, the purpose of using UDP traffic is to see how the delay from the connection between the switch and the controller would affect on its operation like packet loses. This traffic has been generated by using client-server application and the measurement has been done by using different sizes of data. Iperf tool was used to generate this traffic.
The works have been done in a laptop of type MacBook Air with processor 1.4 GHz Intel core i5ntel with 2 cores and 4 GB for the memory. And running Ubuntu 12.04 64-bit LTS Linux Operating System.
The network consists of six OF switches and eight hosts as presented in Fig. 4. The measurements were done in two cases: the first case where the network is idle and the second case where there is background traffic (BT) that is generating new flows during the measurements. The purpose of the BT is to increase the workload for the controllers to see how this affects on its behavior. TCP traffic was used as BT and generated by using Iperf. The packets are transferred from h5 to h7 and from h6 to h8.

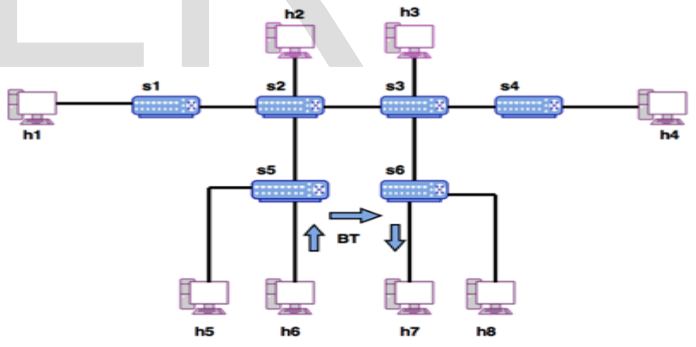


Figure 4 Setup of the network

In the measurements of ICMP traffic, h1 represents the source and h2, h3 and h4 represent the destination respectively. While in the measurements of TCP and UDP traffics, h1 represents the source and h4 represents the destination. The bandwidth was set to 100 Mb/s.
The proposed method to do the enhancement is by using QoS technique, and seeing how this technique affects the performance of the network.
Floodlight controller was chosen to be the tested controller. A QoS module has been added to this controller that does tasks like matching, flow insertion- deletion, classification and policy treating for QoS. This module allows us to define queuing policies and transmit them to a specific queue on switch port with a given action like rate limits.

The QoS was added on the path between s2 and s3 and the workflow for using it as follow:
A. Run the Floodlight controller and connects it with the custom network in Mininet emulator.
B. Create the QoS queues with minimum rate of 20 Mbps and maximum rate of 100 Mbps.
C. The application QoSManger is used to add, modify and delete the policies and services in the network, and enable the controller to start adding the policies and service.
D. The application QoSPath uses the circuitpusher program to push QoS along a path in the network.

## 5 Performance Evaluation

Four OF controllers have been tested to evaluate the performance of the network by using ICMP, TCP and UDP traffics, and the test has been done in two case, first when the network is idle and second when there is a BT in the network.

1. Before the enhancement

For ICMP, by using Ping, RTT has been measured by repeating the test five times and the average was taken as a final result. The RTT is measured for 1, 2 and 3 hops and in the two cases.
Figure 5 shows the time of the first packet without BT and Fig. 6 presents the time with BT.
In the case of no BT, it can be observed that the first packet with Floodlight controller takes more time than in other controllers to reach nearly 15 ms at three hops while in Open-MUL controller, it takes less time to reach nearly 6 ms, thus Open-MUL controller has the best results among other controllers when the network is not overloaded.

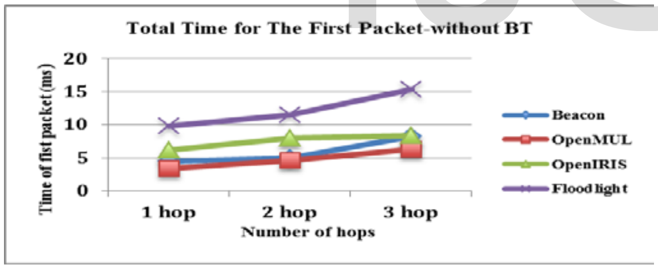


Figure 5 Time of the first packet without BT

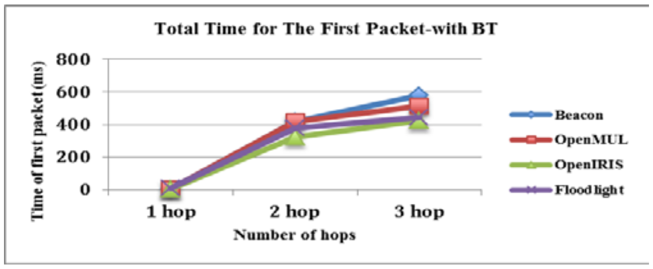


Figure 6 Time of the first packet with BT

In the case of BT, the time of the first packet at one hop is around 12 ms in all the controllers. In two and three hops, the first packet of a flow takes less time in Open-IRIS controller than the others. Thus when the network is overloaded, Open-IRIS controller has the best results of time among other controllers.
Figures 7 and 8 show the average RTT for the controllers in the two cases. There are two observations, the first one; Floodlight controller shows the highest value of average RTT among other controllers in the case of BT and in the case without it, while Open-MUL controller shows the lowest values among other controllers in the two cases.

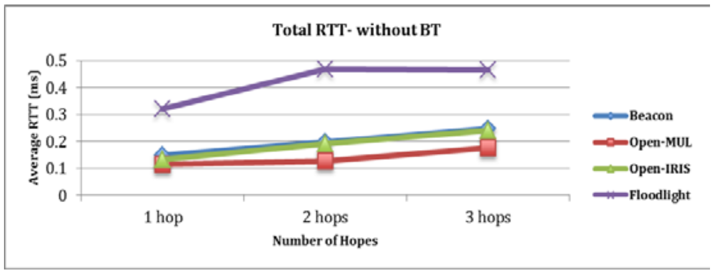


Figure 7 Average RTT without BT

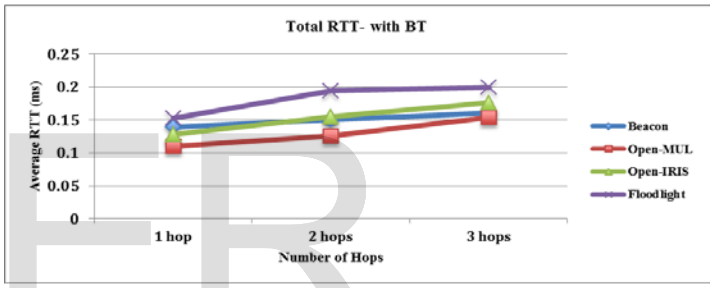


Figure 8 Average RTT with BT

The second observation from these results, the values of average RTT for each controller was decreased when there is BT and this result was expected, since the connection between the switch and the controller is already initiated before beginning the ping and sending Echo Request/Reply packets.
For TCP, the test was done by implementing a TCP client-server application to measure the transfer time of data in different sizes, the measured time includes the time from the establishment of the connection until the close of the connection after sending all the bytes. And this test was done in two cases: without BT and with BT.
The host h1 was configured as TCP client and host h4 was configured as TCP server and the sizes of data are: 100, 200, 350, 500, 750 and 1000 Mbytes. Figure 9 presented the transfer time of the different bytes without BT and Fig. 10 shows the transfer time with BT.
It can be seen when the network is idle, the transfer time of bytes is nearly the same in all the controllers that reach 90 sec to send 1000 Mbytes. While when the network has BT, the transfer time in the controllers is nearly the same when 100, 200 and 350 Mbytes was sent, and when the number of bytes began to increase, Open-MUL controller consumed more time to transfer than other controllers, while Beacon and Open-IRIS controllers have showed the same results almost that reach 230 sec to send 1000 Mbytes.

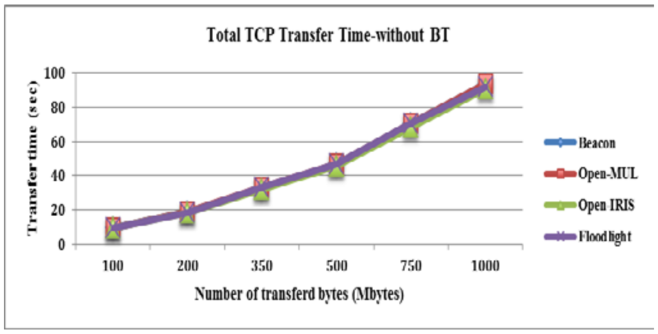


Figure 9 TCP transfer time without BT

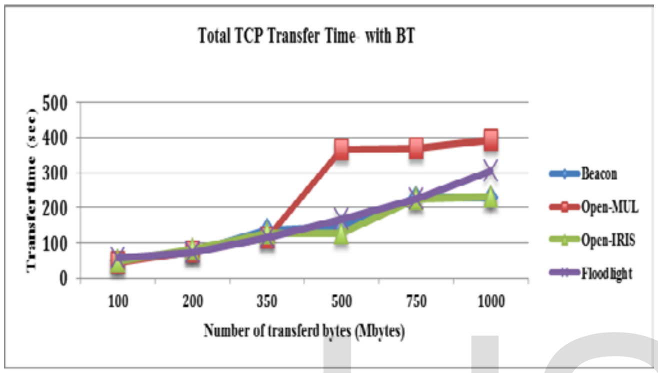
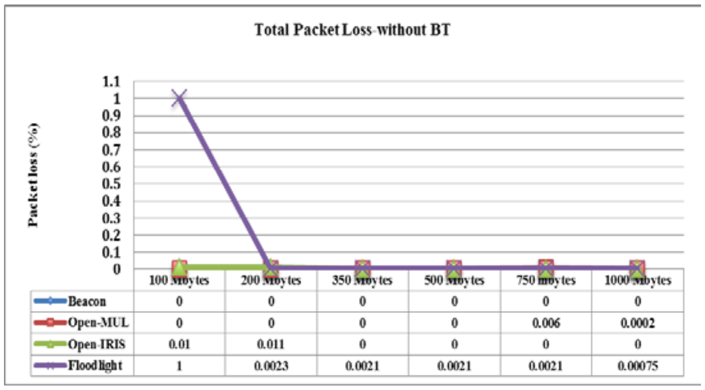


Figure 10 TCP transfer time with BT

For UDP, the test has been done by implementing UDP client-server application and record the percentage of packet loss during sending a data of sizes: 100, 200, 350, 500, 750 and 1000 Mbytes.The host h1 was configured as UDP client and host h4 configured as UDP server.Figures 11 and 12 present the percentage of packet loss in case of no BT and with BT respectively.



| | 100 Mbytes | 200 Mbytes | 350 Mbytes | 500 Mbytes | 750 mbytes | 1000 Mbytes |
|---|---|---|---|---|---|---|
| Beacon | 0 | 0 | 0 | 0 | 0 | 0 |
| Open-MUL | 0 | 0 | 0 | 0 | 0.006 | 0.0002 |
| Open-IRIS | 0.01 | 0.011 | 0 | 0 | 0 | 0 |
| Flood light | 1 | 0.0023 | 0.0021 | 0.0021 | 0.0021 | 0.00075 |

Figure 11 UDP packet loss without BT

From the results, when the network has no BT, the packet loss is around zero except Floodlight that has very small loss when sending 100 Mbytes. While when there is a BT, the controllers show different values of packet loses, Open-MUL controller

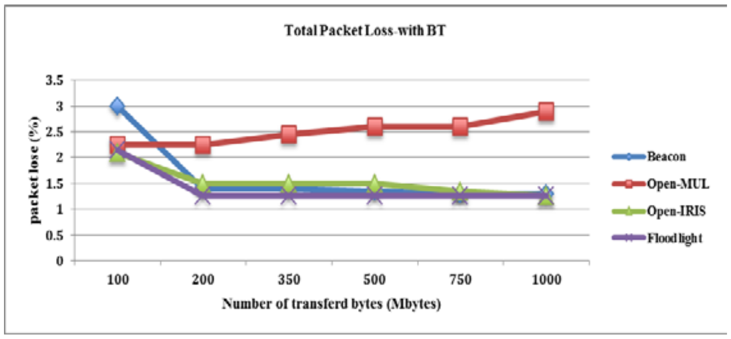


Figure 12 UDP packet loss with BT

shows high values of packet loss according to other controllers, while Floodlight controller was the best one among the other controllers but this value is still considered not good, since the acceptable value for packet loss must be less than 1%.

2. the After the enhancement

The purpose of this section is to see how the measurements for ICMP, TCP and UDP would be after using QoS techniques in the network with Floodlight controller. The obtained results were compared with previous results of Floodlight controller.
For ICMP, after configuring the OF switch with the required policies, BT was started between h5 and h7 and also between h6 and h8 by using iperf. Then ICMP Echo request/Reply was sent from the source to the destination. The source is h1 and the destination would be h2, h3 and h4.
The time for the first packet and the average RTT were evaluated five times and an average was taken as a final result as shown in Fig. 13 and Fig 14. From the results, a significant improvement it can be seen in both the time of the first packet and RTT at 2 and 3 hops.

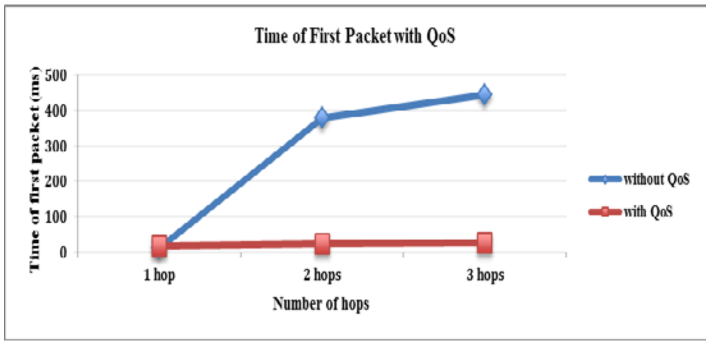
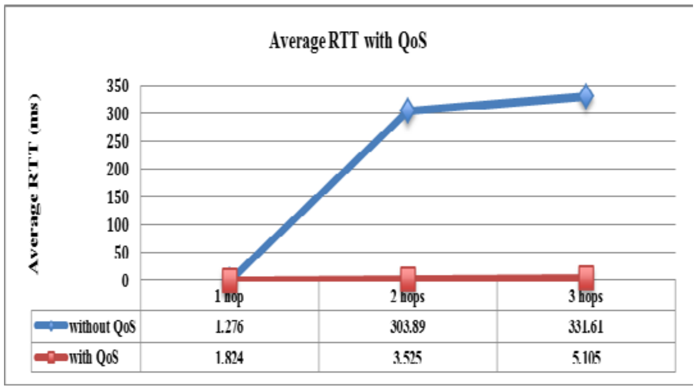


Figure 13 Time of the first packet with QoS

For TCP, the transfer time is evaluated for different data sizes as presented in Fig. 15, it can be observed that the use of QoS technique reduced the transfer time by 68% at sending 100 Mbytes and nearly 50% at sending 200,350 and 500 Mbytes and reduced by 40% at sending 750 and 1000 Mbytes.
For UDP, The UDP differs from TCP that it has no handshake process between switches and controller.At first; the source sends the data to the first switch on the path and the switch



| | 1 hop | 2 hops | 3 hops |
|---|---|---|---|
| without QoS | 1.276 | 303.89 | 331.61 |
| with QoS | 1.824 | 3.525 | 5.105 |

Figure 14 Average RTT with QoS

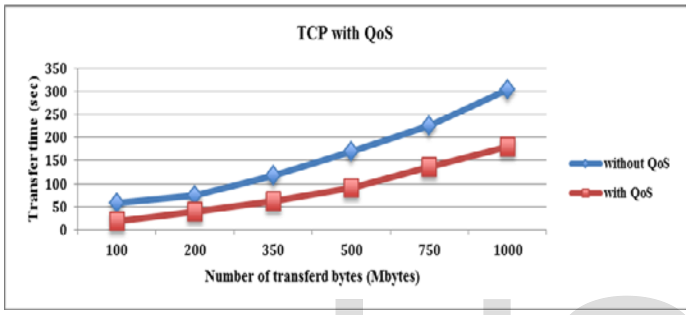
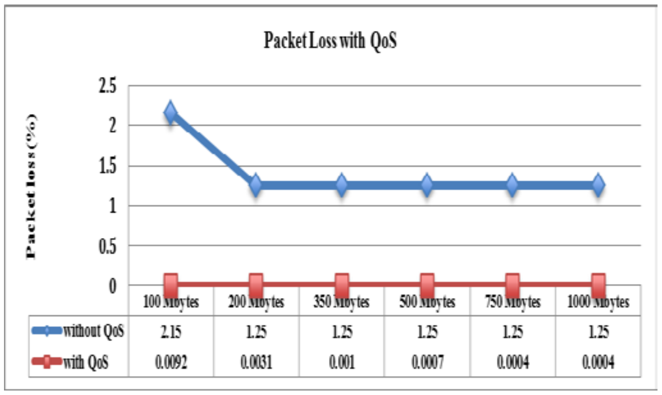


Figure 15 TCP with QoS

has a buffer to store the packets then the packets will be sent to the controller after the buffer becomes full. This will overload the link and if the link did not support this load, this will lead to packet loss.
The other important thing in UDP is the throughput of sending; UDP in general sends datagram one after without retransmitting. This will cause the buffer to be full very fast if the throughput of sending is high, and this will increase the time that the packets should wait until the buffer is empty, and this increases the probability of packet loss.
The results for this UDP measurement are as shown in Fig. 16. From the results, it can be seen distinctly how QoS technique reduced the number of packet loss to 99%.

## 6 Conclusion

The OpenFlow protocol gives a certain degree of freedom in how flows are set up in a network. The measurements of ICMP packets show that the values of RTT for the controllers are decreased when apply a BT in the network and this result was expected since the connection between the switch and the controller is already initiated before sending Echo Request/Reply packets. The Open-MUL controller shows the lowest values for RTT among other controllers in the case of no BT in the network and in the case with BT, while Floodlight controller shows the highest values of RTT in both cases. From the TCP measurements, the transfer time of bytes is nearly the same in all the controllers when the network is idle. While with BT, Open-MUL takes more time than the others to transfer 500 Mbytes and more till 1000 Mbytes.



| | 100 Mbytes | 200 Mbytes | 350 Mbytes | 500 Mbytes | 750 Mbytes | 1000 Mbytes |
|---|---|---|---|---|---|---|
| without QoS | 2.15 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 |
| with QoS | 0.0092 | 0.0031 | 0.001 | 0.0007 | 0.0004 | 0.0004 |

Figure 16 Packet loss with QoS

Form the UDP measurements; Open-MUL controller has large packet loss when the network is overloaded with BT.
From this work, the difference in throughput and delay values for the OF controllers are due to:

1. The used algorithm from the OF controller to distribute the incoming messages between threads.
2. The used mechanisms and libraries for the interaction between the network and the OF controller.

And when QoS have been used, it can be concluded that applying QoS in OF network enhances the performance of Floodlight controller.

## References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "Openflow: Enabling Innovation in Campus Networks", ACM SIGCOMM Computer Communication Review, v.38 n.2, April 2008.
[2] Open Networking Foundation," Open Networking Foundation specifications", Available at: https://www.opennetworking.org/ja/sdn-resources-ja/onf-specifications
[3] M. Jarschel, "An Assessment of Applications and Performance Analysis of Software Defined Networking",University Wurzburg, 2014.
[4] R. Mehra,"Designing and Building a Datacenter Network: An Alternative Approach with OpenFlow", NEC Corporation, 2012.
[5] F. Hu,"Network Innovation through OpenFlow and SDN: Principles and Design", 1st edition, CRC Press, 2014.
[6] CITRIX systems,"SDN 101: An Introduction to Software Defined Networking", 2015.
[7] P. Chhikara, G. Matharu, V. Deep,"Towards OpenFlow Based Software Defined Networks",IEEE ICCIC, Coimbatore, 2014.
[8] Open Networking Foundation, "OpenFlow Switch Specification", version 1.0.0,2009.
[9] Open vSwitch, "Rate-Limiting VM Traffic Using QoS Policing", 2014, available at:
http://openvswitch.org/support/config-cookbooks/qos-rate-limiting/.