Journal of University of Duhok, Vol. 21, No.1 (Pure and Eng. Sciences), Pp 6-14, 2018
DOI: https://doi.org/10.26682/sjuod.2018.21.1.2

THE IMPACT OF DISTANCE BETWEEN THE DATA AND CONTROL PLANE ON OPENFLOW PROTOCOL PERFORMANCE

FARIS **K**ETI*
Dept . of Electrical and Computer Engineering,College of Engineering, University of Duhok, Kurdistan Region-Iraq

(Received: September 11, 2017; Accepted for Publication: May 10, 2018)

ABSTRACT

The Internet has led to the creation of a digital society, where most of things is connected to it. However, in spite of their popular adoption, current internet networks are complex and very difficult to manage. With the increasing complexity of traditional IP networks, Software Defined Networking (SDN) has been emergent as a new norm of networking that can solve many of current internet network management problems. Introducing the concepts of SDN into the network architecture brought the idea of physical separation of these two planes, pushing up the control planes to the centralized controller of the architecture and leaving the data planes remained on the network elements. This paper is to investigate that if the control plane is physically moved away from the data plane, what impact does this have on the performance as seen by users of the network. Also, the effect of this physical distance resulted from the separation of these two planes on the most famous protocol (OpenFlow protocol) performance is conducted.

KEYWORDS: OpenFlow, Data Plane, Control Plane, Software Defined Networking.

I. INTRODUCTION

The community of computer networks researchers has been searching for ideas that enable the use of networks with more programming resources and less need for hardware elements replacement. SDN is an emerging network architecture that virtualizes network infrastructure by separating the control and data plane logic of traditional network devices, creating a flexible, dynamic, automated and manageable architecture [1, 2]. In an SDN architecture, a layer 2 switch forwards packets according to a set of rules that are defined by a software controller. This allows to keep the network device simple and to add functionality to the switch by developing software applications on the controller [3].

This paper is to investigate that if the control plane is physically moved away from the data plane, what impact does this have on the performance of the network. Since, OpenFlow protocol [4] is the most famous open interface protocol that standardizes the communications between the data and control plane, the effect of physical distance resulted from the separation of these two planes on protocol performance is conducted [5].

Section II of this paper is the literature review. Section III discusses the SDN paradigm describing its motivation, network elements that are part of this new structure, in addition, the operation of these components. Section IV describes OpenFlow Protocol which is the interface between control and data plane of the new norm of computer networks known as SDN. Section V presents a brief description of the software used in the implementation of this study. In section VI the effect of distance between control and data planes on the OpenFlow protocol performance is conducted. Section VII concludes the paper.

II. LITERATURE REVIEW

Although SDN is being rapidly deployed and developed, and get too much interest by both artificial and research community, but there have been a small number of studies that focus on the evaluation of SDN architecture performance [3, 5]. Among that few studies, to the best of my knowledge, this paper is one of the first studies to consider the impact of distance between the data and control plane on OpenFlow protocol performance particularly and the SDN architecture in general.

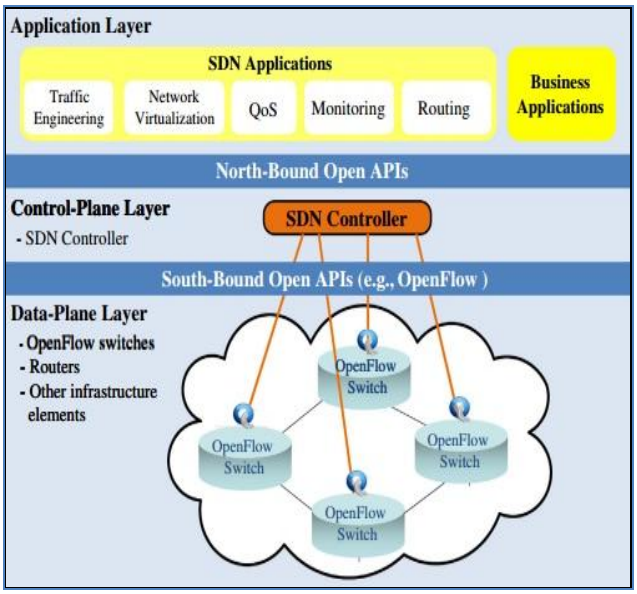
In section VI of this paper a scenario where there is significant delay between the control and data planes has been considered. Multiple tests using different protocols: from ping, TCP file transfer, to UDP traffic streaming. For each test,

* E-mail: faris.alyas@uod.ac

different distances between the control plane and the data plane were experimented. The results show that at the beginning of every communication, there is a mean Round-Trip Time (RTT) approximately three times the programmed delay between the controller and the data plane. This delay significantly affects the OpenFlow protocol performance (increases the time needed by OpenFlow protocol switches to consult the controller) and reduces the throughput especially when transferring small files.

III. SOFTWARE DEFINED NETWORKING (SDN)

SDN is a new network paradigm that decouples the infrastructure (control and data planes) logic of traditional network devices [6].
Figure 1 shows an SDN network architecture.



Fig(1):- Software Defined Network Architecture [7]

In SDN based networks an Application Programming Interface (API) is provided for data plane devices such as switches and routers. API makes the programmability of network devices possible, resulting in a flexible, dynamic, manageable, and automated architecture [8].

There are three main planes in SDN network, which are; data plane, control plane, and management plane. The main SDN planes are shown in Figure 2.

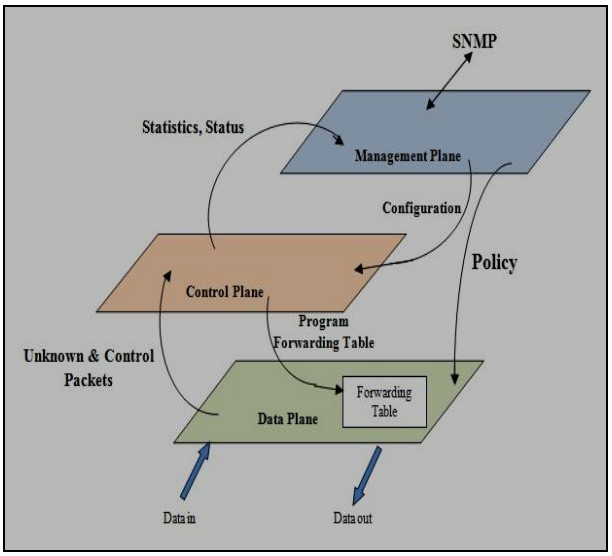


Fig.(2):- The main SDN Planes [9]

The control and data planes communicate with each other via south-band open interface protocol (OpenFlow Protocol), while the management and control planes communicate with each other via north-band open interface.

In the future, SDN will be an important and/or main part of internet technology. As all networking technologies have software to some percentage; therefore, they are all SDN [9].

IV. OPENFLOW PROTOCOL

OpenFlow protocol [4] was proposed to standardize how the centralized software controller and the switches communicate with each other through an open interface protocol between control plane and data plane. The OpenFlow protocol defines the communication between an OpenFlow based SDN controller and an OpenFlow switch. This protocol is what most uniquely identifies OpenFlow technology. At its essence, the protocol consists of a set of messages that are sent from the controller to the switch and a corresponding set of messages that are sent in the opposite direction. Collectively the messages allow the controller to program the switch so as to allow fine-grained control over the switching of user traffic.

OpenFlow switch architecture consists of multiple flow tables in addition to a secure channel which communicates with a controller via OpenFlow protocol [7].
The architecture of OpenFlow Switch is shown in Figure 3.

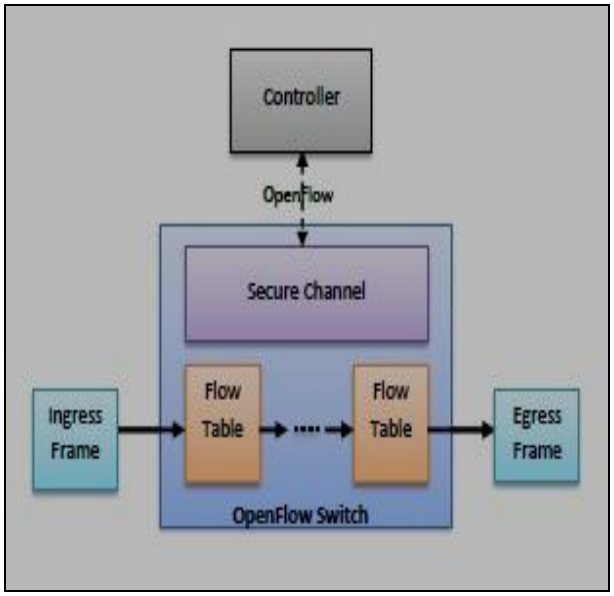


Fig. (3):- OpenFlow Switch [10]

Each flow table consists of a number of forwarding flow entries, each incoming packet matched a correspondent flow entry, then processed and forwarded according to that flow entry. The flow entries of any flow table have the following parameters, which are; matching fields, counters, and a group of instructions. Match fields are used for the process of matching the incoming packets based on previously stored information with incoming packet header, ingress port, and metadata. Counters are used to count up the statistics for every flow such as; the duration of a specific flow, and the number of received packets or received bytes. The instructions are used when there is a match; they determine how to deal with the matched packets [11].
The centralized controller and the switches (Network Elements) communicate with each other through OpenFlow Protocol messages.

Figure 4 shows different types and categories of OpenFlow Protocol messages.

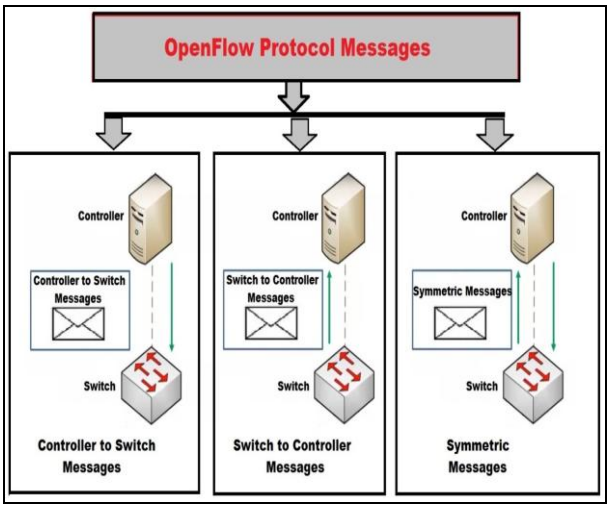


Fig.(4):- OpenFlow Protocol messages

V. SOFTWARE USED IN STUDY IMPLIMENTATION AND TESTS

For the purpose of this study implementation and tests, the softwares in Table 1 are used.

Mininet is the earliest emulation software (Emulator) that provides a simple and easy opportunity to prototype and evaluates SDN topologies, protocols, controllers, and applications [12]. The main property of Mininet emulator is that SDN topologies, protocols, controllers, and applications created, and developed by the emulator can be easily used in a real SDN network without any modification. It is possible to emulate an SDN network with hundreds of hosts by using only a single laptop [13].

The capabilities of Mininet enable students, researchers, and network programmers to prototype SDN networks in an easiest way [12]. But for Mininet to be utilized as one of the powerful tools in emulating the SDN networks, the simulation environment characteristics and qualifications should be considered [14].

Table (1):- Software Used in Implementation and Tests

Software	Function
Oracle VM Virtual Box	Virtualization Software
Linux (Ubuntu)	Host Operating System
Mininet	Network Emulator
POX	SDN Controller Platform
Python	Programming Language
Nttcp	New Test TCP
Iperf	Network Traffic Generator

There are a lot of SDN controllers, however, for the tests of this paper, POX [15] SDN controller is selected because POX began as a controller for OpenFlow protocol, geared towards research and education, and can be used for developing networking software. POX is a software platform developed in Python [16]. It works with all Python versions, and can run under Linux operating systems, Mac operating systems, and Windows operating systems. The core and main modules of POX are developed in python Table 2 contains the names of some controllers and the programming language it supports.

New test TCP program (nttcp) is the software used to measure the transfer rate on a Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or UDP multicast connection.

IPERF [17] is a tool used for network performance measurement. It has client and server functionality, and can create unidirectional or bidirectional streams of data to measure the throughput between two nodes.

Table(2):- SDN Controllers with Appropriated Programming Language

Controller	Programming Language
POX	Python
Ruby	Trema
Beacon	Java
NOX	C++ / Python
OpenDaylight	Python
FloodLight	Java
RYU	Python

VI. PERFORMANCE MEASUREMENTS

In order to evaluate the impact of the location of the SDN control plane with respect to the data plane on OpenFlow protocol performance, and what impact does this have on the performance of the network, a simplified topology was created to emulate different scenarios. The topology (see Figure 5) enables us to change the location of the controller with respect to the data plane, with a constant delay between the hosts.

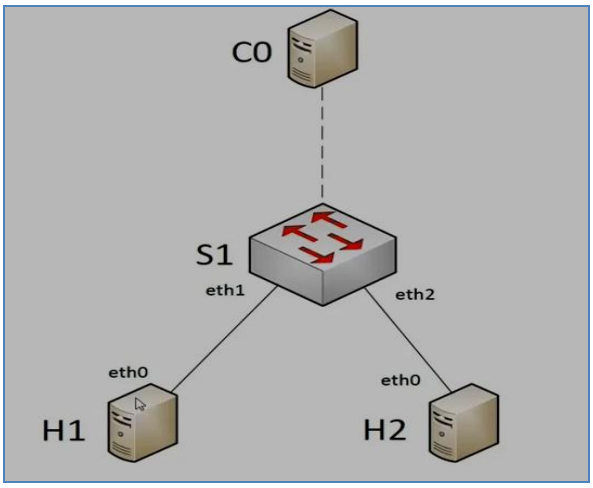


Fig. (5):- Experimental Topology

The topology was created using the Mininet emulator [18]. The following instruction is used for the purpose of creating such topologies by Mininet:

$ sudo mn - - topo Single,2 - - controller = remote

The topology consists of two end devices (hosts), one forwarding device (switch), and the controller. The control plane was implemented using the POX SDN controller [18] which communicated with the OpenFlow switch using OpenFlow protocol.

The geographical location of the controller was simulated by adding a fixed delay to packets arriving and leaving the controller. The first scenario of our experiments was to generate traffic using **ping** with different values of controller delay settings as per Table 3.

Table(3):- Controller Delay Values

SDN Controller Delay	SDN Location
0 (ms)	Reference Controller (Controller in the same location as Data Plane)
15 (ms)	Controller in a different location in the same city as the Data Plane
30 (ms)	Remote Controller (Controller is remote geographically from Data Plane)

The purpose of this scenario from the experiment was to observe the effect of SDN controller delay (controller Round-Trip Time (RTT)) on ping transmission times. It involved sending Internet Control Message Protocol (ICMP) requests from Host 1 to Host 2 repeatedly for many repetitions. It has been observed that the first ICMP request took longer time to complete than all subsequent ICMP requests.

The registered response time for the first ICMP request and all subsequent requests is summarized in Table 4.

It could be noticed, that all subsequent requests for the entire controller delay configurations were completed in an approximately constant average RTT of about 6 ms, indicating that when the SDN switch was completely configured, then the traffic was switched with less considerable delay. This is because the switch's flow table entries had been programmed and updated, therefore; the controller would no longer be consulted by the switch.

Table(4) :_ Controller Delay Values

Delay (ms)	First Ping	Subsequent Pings
	Avg RTT (ms)	Avg RTT (ms)
0 (ms)	27.744	5.47
15 (ms)	44.4	6.23
30 (ms)	92.6	6.41

From Table 4, it can be noted that the RTT taken by the first ICMP request is approximately equal to three times the added delay between the control and data planes.

This denotes that the switch consulted the controller for instructions on how to deal with the new ICMP request three times during the first request. It's expected that for entire flows from Host 1 to Host 2, the first packet will take an additional delay of approximately 3 times the added delay (RTT time) and all subsequent packets will take no extra delay.

The second scenario was the TCP experiment. In this experiment amounts of data where transferred from Host 1 to Host 2 using TCP protocol. As an example usage of TCP files transfer, a Hypertext Transfer Protocol (HTTP) requests were run using the same range of controller delays as per first scenario. The purpose of this scenario from the experiment was to explain the effect of controller location on perceived TCP throughput.

The obtained results plotted in Figure 6.

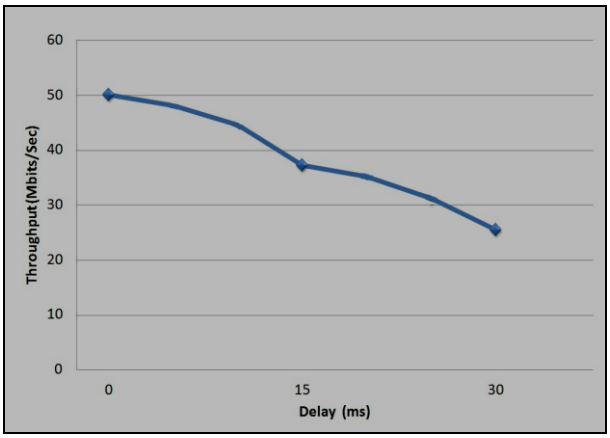
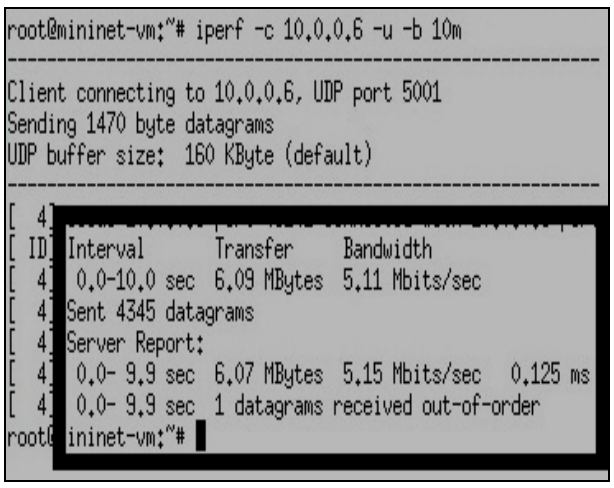


Fig. (6):- perceived TCP throughput**.**

From the results it could be concluded that for particularly small transfers (such as HTTP requests or web page downloads), the distance between the controller and the data planes can have a large impact on the user perceived throughput. Therefore; the location of the SDN controller may need to be carefully considered by network administrators in order to decrease it's heavily impact on network performance.

The third and last scenario from our tests considered the impact of the distance between the control plane and the data plane on the performance of UDP flow traffic. The IPERF [18] tool was used to generate the UDP traffic between the two hosts. The results show that as the distance between the control and data plane increases, the time that the data plane switches took to consult the controller increased. As a result, there is a subsequent increase in the number of packets (datagrams) arriving out-of-order at destination Host, especially at the beginning of each flow (see Figures 7 and 8).



Fig(7):- Less Datagrams Received Out-of-Order

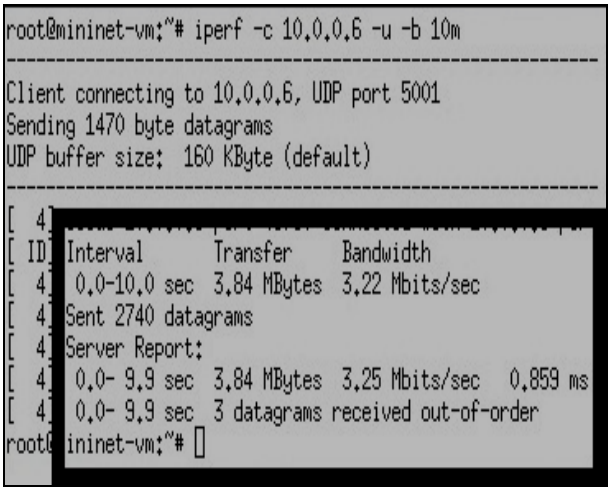


Fig.(8):- More Datagrams Received Out-of-Order

From Figure 7, one can see that when the time taken by data plane switch to consult the SDN controller through OpenFlow protocol was 0.125 ms, then the number of datagrams that received out of order at the destination host was only 1 datagram out of 4345 datagrams have been sent. Or, in other words, the percentage of out of order received datagrams was only 0.023%.

While when the time of messaging between the data plane switch and the SDN controller using OpenFlow protocol standardized messages increases to 0.859 ms as shown in Figure 8 above, then the number of datagrams that received out of order at the destination host become 3 datagrams out of 2740 datagrams have been sent. Or the percentage of out of order received datagrams increased to become 0.11%.

Due to real-time flows nature, upon receipt of the first packet the playback process begins. Therefore; the subsequent packets are buffered for playback because of the playback of the first packet. As such, in addition to the time required for the packet reordering process, the complete real time flow will experience an observed transmission delay. This possible impact should be taken into consideration when the location of an SDN controller determined.

VII. CONCLUSION

For decades to come, SDN is on a way to be an important and permanent part of networking technology. Therefore, this work considered if SDN sees widespread deployment then network operators and administrators might consider remote deployment of controllers to reduce costs. In this paper an analysis of the impacts of distance between the control plane and the data plane on the performance of an OpenFlow based SDN network is performed.

From the results obtained in this paper, it can be noticed clearly that the flexibility offered by SDN can come at the expense of perceived network performance.

For ping traffic, it's clear that the distance between the control plane and the data plane increases the time needed by OpenFlow protocol switches to consult the controller. In response the RTT of the first ping increases by at least three times the delay between the control and data plane. While subsequent pings, however, were fast and not affected at all. For TCP file transfers, it has a large impact on the throughput of transferring small files. While for real-time UDP traffic streaming flows, it could be noticed that a RTT delay in addition to the time required for the packet re-ordering process will be added to the lifetime of the flow.

Therefore, this paper concluded that the network operators and administrators need to carefully consider the placement of SDN controllers within their network. Also the Open Networking Foundation (ONF) who manages the OpenFlow protocol standardization needs to take this impact of distance on protocol performance in to consideration in order to improve the performance or in worst case to decrease the distance impact. This study suggests that the controllers that manage the data plane using OpenFlow Protocol should be placed as close to the data plane as possible.

While, remote controllers could be used for meta-network management, performing tasks such as security or Quality of Services (QoS). This would allow more detailed management to be performed centrally.

REFERENCES

- Guohui Wang, T. S. Eugene Ng, and Anees Shaikh, "Programming your network at run-time for big data applications",HotSDN'12, Helsinki, Finland, August 13, 2012.
- RogérioLeão Santos de Oliveira, and Christiane Marie Schweitzer, et al., "Using Mininet for Emulation and Prototyping Software-Defined Networks", 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Brazil, ©2014 IEEE.
- Seongbok Baik, Chankyou Hwang,and Youngwoo Lee, "SDN-based architecture for end-to-end path provisioning in the mixed circuit and packet network environment", Asia-Pacific Network Operation and Management Symposium (APNOMS), ©IEICE -2014.
- OpenFlow Protocol, The Openflow Switch, [Online].Available: http://www.openflowswitch.org.
- Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy, "Simplifying network management using Software Defined Networking and OpenFlow", 2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS), ©2012 IEEE.
- ONF White Paper, "Software-Defined Networking: The New Norm for Networks," Open Networking Foundation, Tech. Rep., April 2012.
- Ian F. Akyildiz, Ahyoung Lee , and Pu Wang, "A roadmap for traffic engineering in software defined networks", computer networks journal, 2014 Elsevier B.V. All rights reserved ©2014 Elsevier.
- Markus Vahlenkamp, Fabian Schneider, and Dirk Kutscher, "Enabling Information Centric Networking in IP Networks Using SDN", 2013 IEEE International Conference (SDN4FNS), ©2014 IEEE.
- Paul Goransson, and Chuck Black, "Software Defined Networks: A Comprehensive Approach " , Book (1st ed.), 2014 Elsevier Inc. All rights reserved© 2014 Elsevier.
- Alexander Gelberger, Niv Yemini, and Ran Giladi, "Perfor mance Analysis of Software-Defined Networking (SDN)", 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, IEEE Computer Society.
- Thomas D. Nadeau, and Ken Gray, "SDN: Software Defined Net works", Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, First Edition, August 2013.
- B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.
- Diego Kreutz, and Fernando M. V. Ramos, et al., " Software Defined Networking: A Comprehensive Survey", 2015 proceeding of IEEE journal, First Edition ©2015 IEEE.
- Faris Keti, and Shavan Askar, "A New Investigation of Mininet Emulator for Evaluating Software Defined Networks Performance", Journal of University of Duhok, Vol. 18, No.1 (Pure and Eng. Sciences), 2015.
- POX, "Pox SDN controller", [Online].Available: http://www.noxrepo.org/pox/about-pox.
- Python Software Foundation, "Python language reference, version 2.7 " , [Online].Available: http://www.python.org,.
- Iperf network traffic generator, [Online].Available: (https://iperf.fr/).
- Mininet: An Instant Virtual Network on your Laptop. [Online]Available: http://mininet.org.