Improvement of Multiprotocol Label Switching Network's Performance using Software Defined Network Approach

Fatima LAASSIRI[1], Mohamed MOUGHIT[2], Noureddine IDBOUFKER[3]

Abstract— Software defined networking (SDN) combines approaches of connection oriented technologies and routing overlay technology presenting a new open and programmable network. Decoupling control and data plane from network devices and bringing control plane in logically centralized SDN controller which can be then act as a network operating system.

This article presents the improvement of the performance of the quality of service (QoS) parameters which are: end-to-end delay, latency, jitter, lost packets, and Mean Opinion Score (MOS), under Multiprotocol Label Switching networks. (MPLS) with a contribution of adding a SDN for MPLS to improve the quality of services on MPLS under Omnet 4.6 ++, by the development of a new program in C ++ that allows to integrate an SDN controller for the centralization of all traffic in the MPLS architecture.

The realization of this work relies on an IPv6 addressing scheme for all MPLS architecture with and without SDN, and for SDN operations, it is based on the OpenFlow protocol.

Index Terms—SDN, MPLS, QoS, OpenFlow protocol, Omnet, IPv6, SDN Controller.

1 Introduction

Software Defined Network (SDN) [1] is a new network paradigm used to simplify network management, reducing the complexity of network technology.

This work aims at changing the amelioration of the nature of the performances at the QoS parameters (End-to-end delay, latency, jitter, lost packets, and Mean Opinion Score (MOS)), under MPLS with a input the SDN for MPLS using Omnet 4.6 ++, by developing than the new program with C++ that permits to incorporate a SDN controller for the centralization of all traffic in MPLS architecture, the nodes use an IPv6 addressing scheme all MPLS architecture with and without SDN.

2 Problem and solution

Traffic is continuously expanding and a growing variety of services is flowing through by the Wide Area Network (WAN) infrastructure. It increases the need for a mechanism that can manage resources in a more efficient and dynamic way. MPLS traffic engineering has been around for many years, but the distribution of traffic engineering decisions has led to inefficiencies. Now, the greater availability and implementation of OpenFlow protocol solves the problem via the ability to enable centralized intelligent controllers (SDN) to complete the distributed control plan of the network, adding a process of vision and decision-making at the network level. By taking advantage of OpenFlow switches with its large memory for more efficient execution in order to coordinate layer 2 addresses and OpenFlow headers. In this way: Better ability to make good resource allocation decisions for path creation and optimization, and search for backup resources, etc.

3 State of the Art

The Ericsson Research Center [8] is currently working to support MPLS with the OpenFlow protocol. The goal being to transform a switch with Openflow functionality enabled, MPLS switch aims to introduce many services faster, easier and cheaper.

3.1. Multiprotocol Label Switching Concepts and Architecture

MPLS is a mechanism for transferring data on label switching, when a packet enters an MPLS domain, a tag is attached to the frame. It is inserted at the entrance of the MPLS network and removed at its output, this insertion takes place between the data link layer (Layer 2) for switching and the network layer (Layer 3) for routing to form a network IP address with a high level of performance to transport protocols such as IP. MPLS unified data transport for customers using a packet switching technique. It can be used to carry any type of traffic, for example voice or IPv4 packets, IPv6 and even Ethernet frames. The goal of MPLS is to guarantee speed, traffic engineering, and QoS. The MPLS domain has two main types of switches, namely; the MPLS edge switches, which mainly consists of (Label Edge Routers) LER [9] and the MPLS kernel switches that they essentially reside in Label Switch Routers (LSR) [10].

4 SDN Architecture Concepts and ARCHITECTURE

SDN enables networks to dynamically react to changes in usage patterns and network resource availability. Network architectures can be added instantly, and respond to user

International Journal of Scientific & Engineering Research, Volume 8, Issue 1, June-2018
ISSN 2229-5518

requests, SDN provides a separation between the control plane functions (Controller) and the data plane (Switch) networks using a protocol that modifies the transfer tables in the network switches. This optimizes networks on the fly and responds quickly to changes in network usage without the need to manually reconfigure existing infrastructure or purchase new hardware. He separates a control of network devices from the data they carry and the actual network hardware switch software, the controller has a complete view of the entire network and its state, with which switches (Network Resources) and applications (Network Consumers) can communicate in real time. It allows networks to interact with applications and reconfigure efficiently as needed, allowing it to implement multiple logical network topologies on a single common network.

4. 1 The Open Flow protocol

The Open Flow protocol is a multivendor standard defined by the Open Networking Foundation (ONF) [11] for the implementation of SDN in network equipment.

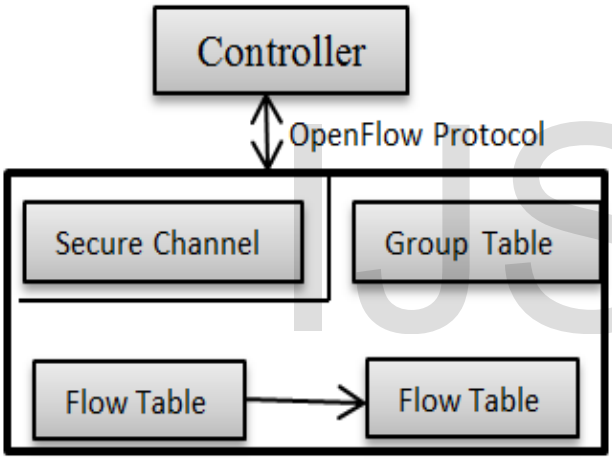

Fig.1.The OpenFlow protocol

It defines the interface between the controller and the OpenFlow switch. It allows the controller to tell the switch how to handle incoming /outgoing data packets. See Figure 1 and 2 below.

The OpenFlow switch can be programmed to:

--Identify and classify packets from an input port based on various packet header fields;

--Process packages in a variety of ways, including modifying the header;

--Drop or push packets to a particular output port or to the OpenFlow Controller.

OpenFlow instructions are passed from a controller to a switch in the manner of "flows", each individual flow contains packet matching fields, flow priority, various counters, packet processing instructions, delays flow. They are organized into tables. An incoming packet can be processed by streams in multiple "pipelined" tables before leaving on an output port.

The OpenFlow network architecture consists of three layers:

--One or more virtual and / or physical OpenFlow switches;
--One or two OpenFlow controller (s);
--One or more OpenFlow applications. For an illustration, see Figure 2 below.

The OpenFlow controller maintains the OpenFlow protocol communication channels to the switches, exposing a Northbound API to OpenFlow applications. The Northbound API can be thought of as an abstraction of the network and allows OpenFlow applications to read the state of the network and order the network to perform various tasks [12].

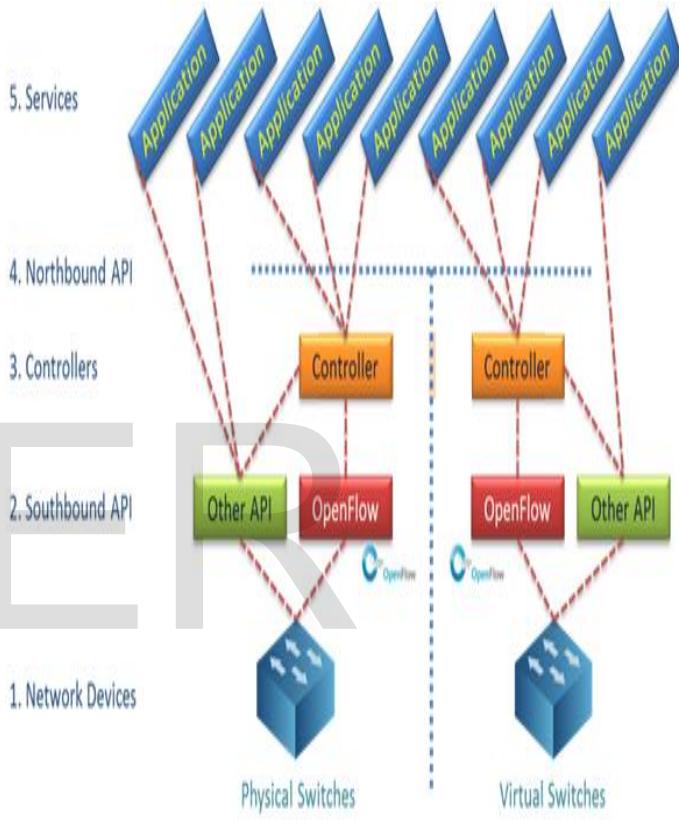

Fig.2. OpenFlow Architecture

5 Methods for the implementation of MPLS Simulation with and without SDN under Omnet 4.6++

The contribution of this work is done through the development of a new program in C ++ that allows to integrate an SDN controller with the MPLS architecture. This simulation are implemented under Omnet4.6 ++, with the first displaying an MPLS network only without SDN (Fig 3) and the second offering the addition of the program SDN for the MPLS network (Fig 4) that it is based on the implementation of OpenFlow protocol. It content the grouped into six packages, like, it is described in the following modules (OpenFlow, ControllerApps, HostApps, HyperFlow, Kandoo, Utility)

In this paper MPLS technology is implemented on the metro political area in the workspace at two scenarios

International Journal of Scientific & Engineering Research, Volume 8, Issue 1, June-2018
ISSN 2229-5518

The simulation is done with the help of OMNET4.6++ and the simulated environment is as shown in the below screenshots.

Scenario I: MPLS is implemented with 6 LSR routers, 2 LER with the first is IngressLER Router and the second is the EgressLER Router.

Scenario II: MPLS is implemented with 6 LSR routers, 2 LER with the first is IngressLER Router, the second is the EgressLER Router, and the addition the SDN_Controller with a switch_OpenFlow.

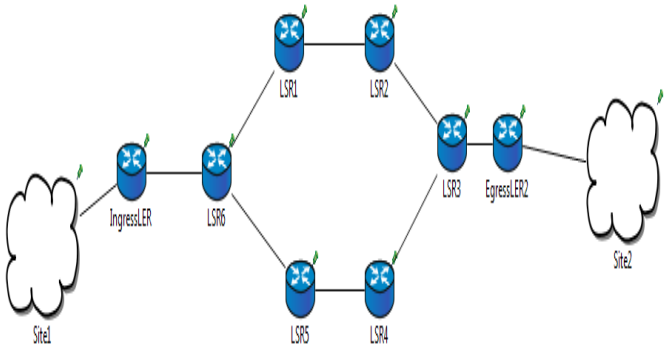

Fig .3. Scenarios 1: MPLS network architecture without SDN.

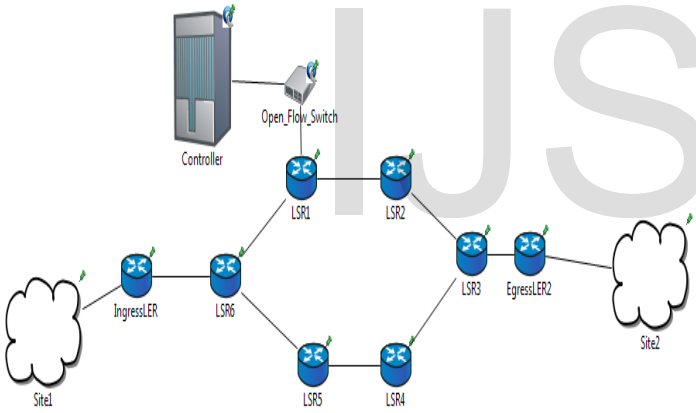

Fig .4. Scenarios 2: MPLS network architecture with SDN.

The OpenFlow switch communicated the controller which packets are unrecognized. So that the latter decides the way, it controls the MPLS tags and it installs the principles to send packets with the corresponding MPLS labels along the way, this is updated by the OpenFlow protocol, with a central controller programming flow.

A data plane MPLS node implements three header modification operations: [13]

Push: A new label on the MPLS stacks or, if there is no stack present, it inserts a label to form a new stack.

In	VALN	Ethernet			MPLS		IP			Transport	
Port	ID	SA	DA	Type	Label2	Label1	SA	DA	Proto	Src	Dst

Fig. 5 OpenFlow Twelve tuple for MPLS rules

Pop: Burst the top label of the stack.

Swap: Swap the top label on the stack for a new label.

The MPLS label stack is inserted between the IP and MAC headers (layer 3 and layer 2) of the package. MPLS label stack entries are 32-bit, 20 of which form the actual label used for the transfer. The other bits indicate the QoS treatment, the top of the stack and the duration of its life.

MPLS push and pop actions, they rewrite the header by inserting fields. Rather than inserting MPLS actions into the OpenFlow Open Process Packet Pipeline (Fig. 5), they can be included in the actions of the flow table, just like physical ports and virtual ports they are bundled with physical ports in a virtual port table. Each virtual port table row contains entries for the port number, the parent port, the actions to be performed by the virtual port, and the statistics.

The MPLS actions in the virtual port table are as follows:

push_mpls: Push a 32-bit label at the top of the MPLS tag stack and copy the TTL and QoS bits from the previous IP header or MPLS tag,

pop_mpls: Place the top label on the MPLS stack and copy the TTL and QoS bits to IP header or old MPLS tag.

swap_mpls: Exchange the 20-bit transfer label over the MPLS stack.

decrement_ttl: Decrease the TTL and drop the package if it has expired.

copy_bits: Copy the TTL and QoS bits from the IP header or the MPLS label and add a counter to the OpenFlow statistics that it increments each time as a virtual port, and it drops a packet because of the expiration of the service life.

The OpenFlow protocol has been extended with the following messages to allow the controller to program label switching paths (LSPs) in the switches:

vport_mod: Add or delete a virtual port number with parameters such as: parent port number, virtual port number.

vport_table_stats: Send statistics for the virtual port table. The statistics include the maximum virtual ports supported by the switch, the number of virtual ports used, the number of searches, the number of port matches and the number of string matches.

port_stats: OpenFlow port_stats message applies to virtual ports, but only the fields tx_bytes and tx_packets are used. Finally, the OpenFlow message switch_features_reply has been modified to include a bit indicating whether the switch supports virtual ports.

6 Results and Discussion of Simulation in Quality of Service Criteria (QoS)

The parameters of the QoS are defined and simulation is done with Network Simulator OMNET++ and the results obtained are discussed in this section. The various metrics that we are calculating are

-End-to-end delay
-Latency
-Jitter
-Number of lost packets
-MoS

6.1End-to-end delay under MPLS with and without

International Journal of Scientific & Engineering Research, Volume 8, Issue 1, June-2018
ISSN 2229-5518

SDN

Fig. 6. present the the end-to-end delay results, that the graph shows the MPLS free scenario with a higher value of (56 ms) compared to the MPLS based SDN scenario that, it has a reliable delay of (33 ms), the one that it justifies as SDN adds a positive appreciation for MPLS.

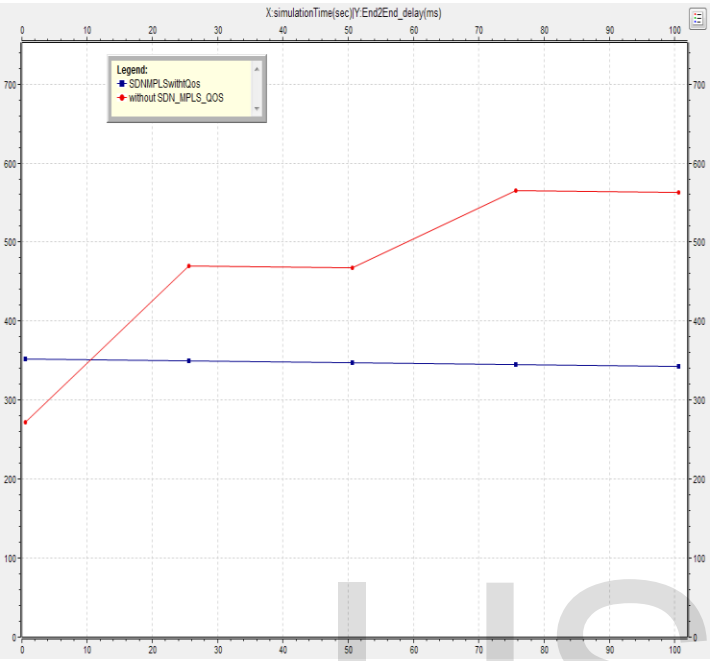

Fig.6. End-to-end delay under MPLS with and without SDN.

6.1 Jitter under MPLS with and without SDN.

The Fig.7 illustrates jitter, Where the MPLS scenario is based on SDN is about 33 ms, that it is half of MPLS without SDN that, it has the value of 31 ms, which results that the addition of a SDN network for MPLS is very low to compared with the MPLS without SDN.

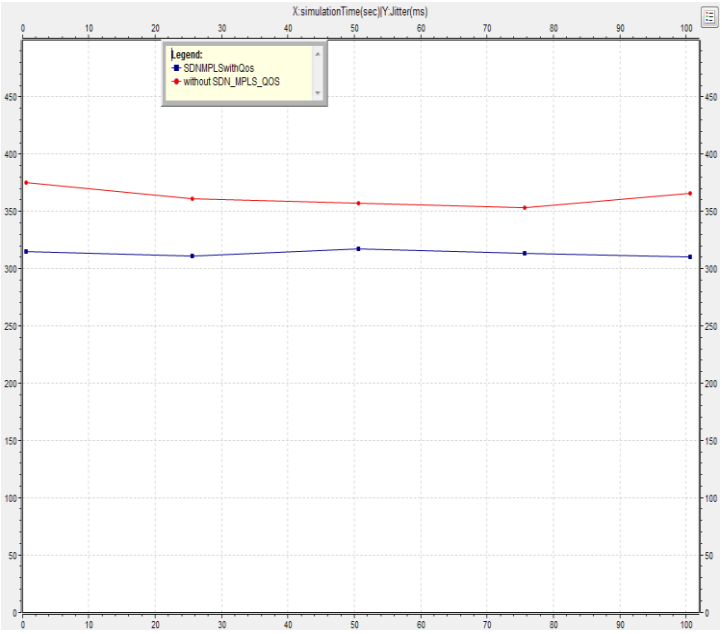

Fig.7. Jitter under MPLS with and without SDN

6.3 Latency under MPLS with and without SDN.

Fig. 8. shows evolution of latency in MPLS network without SDN is less (50 ms) but using SDN latency decrease to 30 ms.

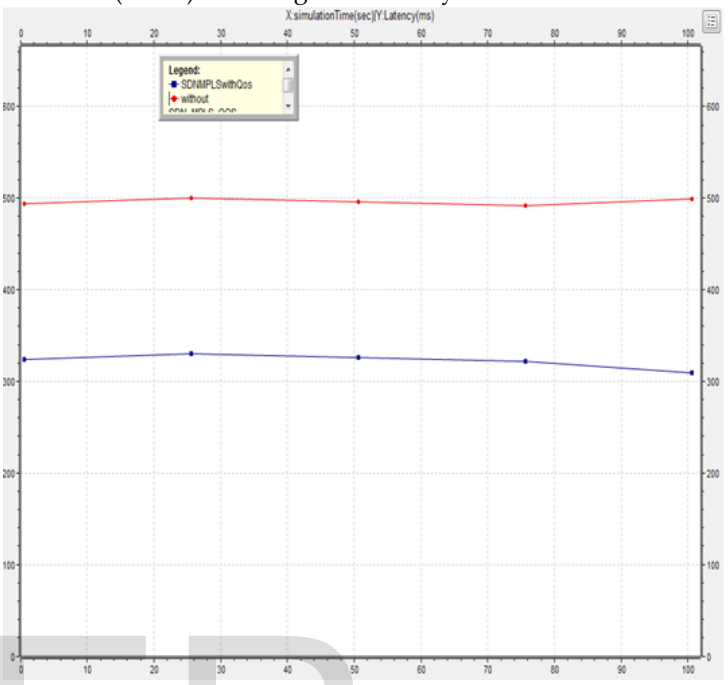

Fig.8. Latency under MPLS with and without SDN.

6.4 Lost packages under MPLS with and without SDN.

Fig.9. watch the MPLS without SDN is 22%, so it is very high, compared to the MPLS approach with SDN, it is about 14%. This shows the impact of adding SDN view to the MPLS network.

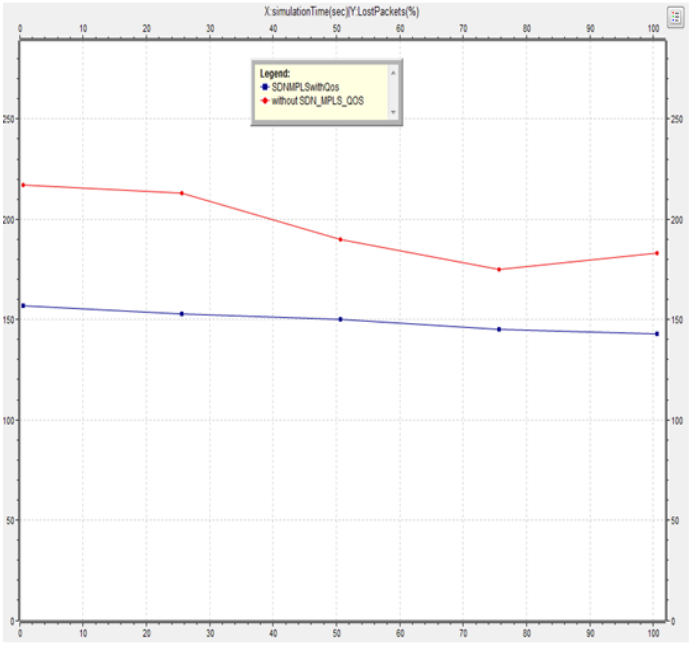

Fig.9. Lost packets under MPLS with and without SDN.

International Journal of Scientific & Engineering Research, Volume 8, Issue 1, June-2018
ISSN 2229-5518

6.5 MOS under MPLS with and without SDN.

Fig.10. shows that the MOS offered by the MPLS approach without SDN is 2, whereas the approach based on the MPLS with SDN is about 3, which it presents an indicator of the increases in the quality of the vocal transmission.

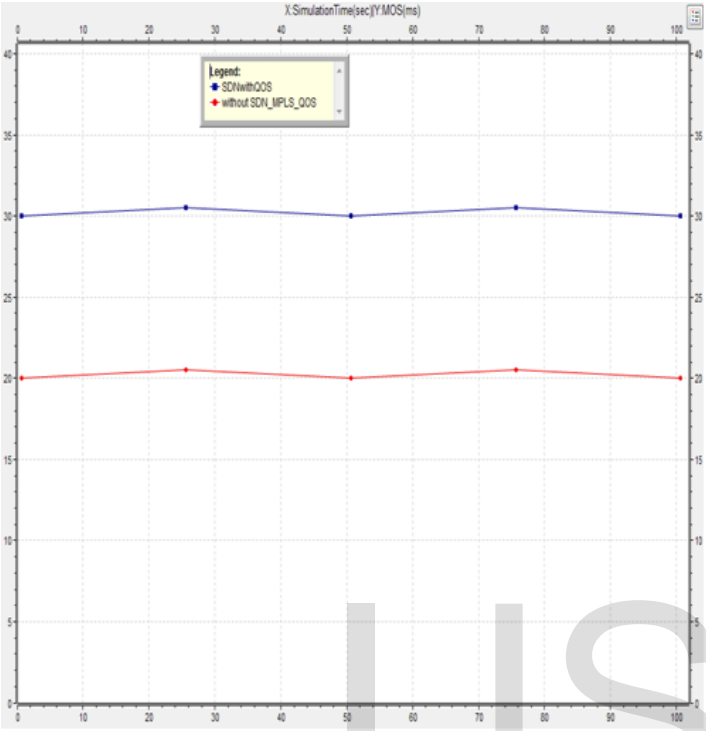

Fig.10. MOS under MPLS with and without SDN.

7 CONCLUSION

This paper has considered two scenarios in MPLS technology with Software Defined Network in the criteria of amelioration of the performance of quality of service. The simulation is to apply the environment with the help of OMNET4.6++. It has been concluded that the OpenFlow protocol implementation and the addition the SDN controller for MPLS have a high impact of QoS parameters. This summarizes that MPLS with SDN is more preferable than MPLS without SDN.

REFERENCES

[1] Fatima LAASSIRI, Mohamed MOUGHIT, Noureddine IDBOUFKER, "Evaluation of the QoS parameters in different SDN architecture using Omnet 4.6++", IEEE, December 2017.

[2] Andreas Lobinger, Szymon Stefanski, Thomas Jansen, "Coordinating Handover Parameter Optimization and Load Balancing in LTE Self-Optimizing Network", July 2011.

[3] Bakouk Mohamed, Khamlichi Idrissi Youness, Moughit Mohamed, "Taking account of trust when adopting cloud computing architecture", IEEE, February 2017.

[4] O. Akinsipe, F. Goodarzi, M. Li "Comparison of IP, MPLS and MPLS RSVP-TE Networks using OPNET", International Journal of Computer Applications (0975 – 8887) Volume 58– No.2, November 2012.

[5] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, "Are we ready for SDN? Implementation challenges for software-defined networks", IEEE, 12 July 2013.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. HandleyE. Schooler, "SIP: Session Initiation Protocol RFC 3261, JUNE 2002.

[7] Joo-Chul Lee, Jung-Soo Park, "Fast Handover for Proxy Mobile IPv6 based on 802.11 Networks", April 2008.

[8] .https://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2009-ttnfa2010/cloarec-ferreira/tableflux.html, January 2018.

[9] Dr. Reyadh Shaker Naoum, Mohanand Maswady, "Performance Evaluation for VOIP over IP and MPLS", WCSIT Journal 2012.

[10] L.S.R. ArambewelaaL.D.A.M. ArawwawalaaW.D.Ratnasooriya, Antidiabetic activities of aqueous and ethanolic extracts of Piper betle leaves in rats", November 2005.

[11] Open Networking Foundation, "ONF SDN Evolution", September 2016.

[12] https://noviflow.com/the-basics-of-sdn-and-the-openflow-network-architecture/?gclid=CjwKCAiAhfzSBRBTEiwAN-ysWH97m1siNHZCcuDxyY5uYNhlYyf-2CrzQEw6qcmfhv3pxxaqQk1ibRoCoR0QAvD_BwE,October 2013.

[13] James Kempf, Scott Whyte, Jonathan Ellithorpe, Peyman Kazemian, Mart Haitjema, Neda Beheshti, Stephen Stuart, Howard Green, "OpenFlow MPLS and the Open Source Label Switched Router", This paper was peer reviewed by subject matter experts for publication in the Proceedings of ITC 2011.

AUTHOR DETAILS:

Corresponding author :

Fatima LAASSIRI[1]: IR2M Laboratory, FST, Univ Hassan UH1- Settat, Morocco, laassiri.fati@gmail.com

Mohamed MOUGHIT[2] : IR2M Laboratory, FST, Univ Hassan 1 UH1- Settat, Morocco
EEA&TI Laboratory, FST, Univ Hassan, Mohammedia, Morocco
National Schools of Applied Sciences Khouribga, Univ Hassan 1, UH1- Settat, Morocco

Noureddine IDBOUFKER[3]: National School of Applied Sciences, Univ Cadi Ayyad Marrakech, Morocco