PJET, Vol. 1, No. 2, June 2023: 43-56

Poly Journal of Engineering and Technology

j o u r n a l h o m e p a g e : https://journals.bdu.edu.et/index.php/PJET/index

ORGINAL ARTICLE

Integrating MPLS on the Data Plane to Enhance Throughput and Response Time of SDN

Belayneh Teshome Kebie[1], Mekuanint Agegnehu Bitew[1]

[1]Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar University, Bahir Dar, Ethiopia

ABSTRACT

Software defined network (SDN) is an emerging technology that enables open networking. Recently, many works have been done to improve the performance of SDN systems. However, there are gaps in integrating multiprotocol label switching (MPLS) on the data plane. Implementing MPLS on the data plane will help us to handle small group of packet flow without the involvement of the controller. The aim of this paper is to implement MPLS on the data plane and investigate throughput, response time and the effect of type of switch on the performance of the SDN. We employed policy-based routing and MPLS with distributed controllers to avoid single point of failure and manage traffics. We classified users as low-level, medium-level, or high-level users, and then we let MPLS handle low-level packet flows, while using policy-based routing to handle the remaining high-level and medium-level packet flows. The proposed scheme reduced the response time to 0.2ms, and it enhanced the overall throughput. Based on the simulation results, we concluded that integrating MPLS with SDN is an efficient method to improve the overall performance of SDN.

***Key words**: Software defined network, multiprotocol label switching, policy-based routing, throughput, response time.*

DOI: https://doi.org/10.20372/pjet.v1i2.1510

Corresponding Author:
Mekuanint Agegnehu Bitew, Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar University, Bahir Dar, Ethiopia
Email: memekuanint@gmail.com

1. Introduction

The desire of having programmable network began in 1990s; two groups actively involved namely, open signaling and active networking group [1]. SDN is a network that is primarily made up of virtual devices and managed by software. The concept of SDN is introduced in 2008 [2] with the idea of programmable data plane**.** In traditional networks, data plane and control plane are tightly coupled which makes the network management task difficult. However, in SDN, control plane and data plane are loosely coupled which makes it preferable over traditional network [3]. The control plane is responsible for managing the entire network, and the data plane is responsible for flow of information and storage. The principle in SDN is the network managed by the software [2]. Since its origin, it comes with many opportunities, like, internet of things (IoT), virtual private network (VPN) and cloud computing. SDN also eliminates the need of firewalls and Intrusion Detection Systems (IDS) in the Network topology [4]. Recently, SDN is becoming popular and applicable in many organizations. Many works have been done on the performance of SDN, however, there is still a gap in considering the integration of MPLS on the data plane to let the data plane itself handle low-level packet flows called mice flows [1]. To the best of our knowledge, there are no works on integrating MPLS on the data plane and making a hybrid data plane. For example, articles [5], [6], [7] and [8], did not consider integrating MPLS on the data plane to reduce the load of the controller. Our paper focuses on investigating throughput and response time aspects of SDN by integrating MPLS on the data plane with policy-based routing and investigating the impact of the type of switch on the performance itself. The motive behind this work is the need for new approach to address performance bottlenecks of SDN. Here, we integrated MPLS on the data plane making hybrid data plane. In our approach, packet processing is taken place on both the data plane and the control plane by classifying packets as low-level, medium-level and high-level packets. We enabled MPLS on the data plane to process low-level packet flows while the remaining medium-level and high-level packets are being processed by the control plane by the help of standalone switches [9]. In our experiments, we took different scenarios, such as size of data, number of hops and link failure into consideration. We also used different types of switches to investigate their difference in terms of response time for different group of packets since the operating systems they operate are different.

1. RELATED WORKS

Many works have been done on improving the performance of SDN by just focusing on some selected parameters and specific scenarios. Among those works, we presented, the following ones due to the fact that they are closer to this work. [7], have done their research work with the aim of enhancing the performance of SDN; they have used mininet emulator to create topology, then they developed QoS module using python.

In their network diagram, there are three different networks and there is one central controller that interconnects those three different networks, then they compared the performance of the module with other modules in terms of throughput and response time. Following their findings, they claimed that their module demonstrated remarkable result compared with NOX, ROIA, and other packet schedulers, however, their module sometimes achieved lower, but they didn't explain why? [6], has done a work entitled "SDN-Based Mechanisms for Provisioning Quality of Service to Selected Network Flows" through status monitoring and QoS-based path setup modules developed using floodlight controller. As a first measure, he grouped the packet flows into three groups, as critical with QoS, with no QoS, and QoS non critical, then he observed that the throughput of the critical flow was 9.5mbps. But he didn't consider the inter-dependency between QoS metrics and even the average throughput is not good enough. [3], studied latency that Openvswitches experience while setting flow path. For their investigation, the authors identified two scenarios i.e. the first one is reducing the update time of every switch, however, openvswitches do have minimal updating time, this implies it will not have significant impact in flow setup. The second approach is reducing the number of switches updating at a time during path setup, this lets the authors manage flows in such a way that performance requirement can be met. The authors tested their model using different topologies, and then they stated that their approach reduced the flow setup latency at end users' aspects, however, the authors did not take all types of traffic flows into account, even the approaches that they used to reduce the number of switches updating simultaneously during path setup is imposing increasing number of error rates in the network that results in increasing packet loss. SDN can be deployed in MPLS network to further improve the performance of the entire network. There is also a work on hybrid network [5], have proposed an overlay method to use OpenFlow controller in optical packet switched network; their key idea is separating the controlling function of the network in to two i.e. on the top, there is an OpenFlow controller that is capable of controlling the entire network, and then in the middle of the network, there is GMPLS that controls the optical packet switched network. This approach could potentially reduce the load of OpenFlow controller, but still, there would be significant delay as the number of hopes increases, [8], have proposed a method to incorporate MPLS in OpenFlow by extending OpenFlow tuple from ten to twelve where the two fields are used to store MPLS information, such as TTL push and pop labels. The authors observed that their model increased the throughput significantly compared with traditional MPLS network. However, there is a gap in considering response time and other metrics. [10], used both SDN and MPLS to improve resource allocation and to study the interdependency between flows. In their approach, they have identified two resource re-allocator modules namely flow level resource re-allocator and LSP level resource re-allocator. Flow level resource re-allocator is used to avoid congestion by rerouting the flow whenever there is overflow in the link from predefined threshold. And the LSP level resource re-allocator is used to control the flow whenever the flow level resource re-allocator is unable to control congestion. Their approach improves resource utilization

and throughput; However, they did not consider the packet loss during rerouting. [11], has implemented MPLS in SDN enabling the controller to assign labels to the packets when the edge switches request for labels. In his approach, the edge switches forward packets to the core switches using labels assigned by the controller. The controller is also responsible to install the path on core switches. The author claimed his approach achieved high hit rate to fill flow rules in the entire topology. However, this approach causes significant load on the controller.

[12], have implemented MPLS and SDN hybridization where all the decisions are still made by the controller i.e., the controller is responsible to install labels to the packets and establish the label switched path (LSP) between the edge and the core switches. The authors' ultimate goal is to boost the performance of MPLS network letting the centralized controller monitors the entire network infrastructure that implies resource allocation, network discovery and establishing the LSP needs to be taken over by the controller; this often helps the performance of the network be enhanced, however, the controller becomes more burdened and be affected by single point of failure since establishing path, assigning labels and resource allocation is still the task of the controller although the MPLS helps the controller by looking after packets on the basis the labels assigned by the controller.

2. MATERIALS AND METHODS

Our proposed method is capable of handling packets on both the data plane and the control plane where low-level packets are handled on the data plane through MPLS and the remaining high level or medium level packets are handled on the control plane using policy-based routing implemented on distributed controllers. We classify users as low-level, medium-level and high-level based on their VLAN prefix. The VLAN prefixes are 1, 2 and 3 for High-level, medium-level and low-level users respectively. The traffic flow from high-level to low-level users and from medium-level to low-level users is taken over by the controllers; in this case, the controllers are required to establish paths and forward packets along with the best alternative paths between two communicating end points.

However, if the traffic flow is in the opposite direction that is from low-level to high-level and from low-level to medium-level, the MPLS is responsible to route the flows along with the best alternative path. MPLS needs to establish paths between two end points, then it handles errors occurred, waiting, forwarding and packet scheduling. In addition, the controllers forward the packets through the path other than the best alternative path; that is what we call policy-based routing letting the traffic flow be balanced across the entire network while improving network resource utilization. On the basis of their priority, we are able to handle packets in such a way that performance goals can be met. The proposed method is summarized diagrammatically in figure 1.

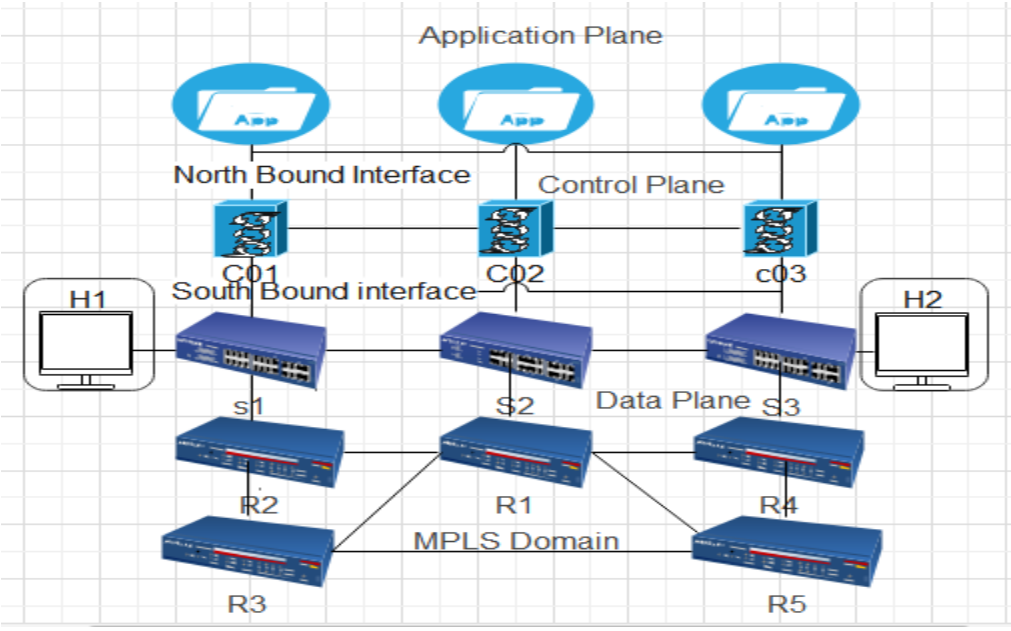

Figure 1: Proposed Method Architecture

3. 1. Topology and Simulation Scenarios

We created a network topology containing MPLS core network integrated with SDN. MPLS is used to process the low-level packets on the data plane without the involvement of the controllers while the controllers are responsible for top to bottom flows. We used distributed controllers where the controllers are distributed physically to let them have equal burden with global view, therefore, they share the loads fairly in such a way that the performance of the network can be improved. The controllers are installed on virtual machines such ODL VM and ONOS VM, then NAT ensures communication between logical devices and VMs. Mininet is used to create topologies integrated with GNS3, then we added the interfaces of routers to openvswitches interfaces so as to establish the communication between the core MPLS and mininet topology. Mininet is an open source simulation software designed for simulating SDN. It is developed using python programming language that is user friendly and able to connect thousands of

devices with simple command lines and APIs [2], [13]. GNS3 is an open source graphical network simulator that connects many devices either in a single or multiple servers and cloud infrastructure. The interface of the routers connected to openvswitches is managed by the OpenFlow controller while they are sending and receiving the packet. We have two kinds of topologies 1) the topology inside GNS3 environment and 2) the topology inside mininet. Inside Mininet, we used 32 hosts connected to different openvswitches, and those switches are connected with each controller node. We have used four controllers through physical controllers' distribution scheme where each controller is connected with every openvswitch to avoid single point of failure. Inside GNS3 topology, there are five legacy routers connected with mininet virtual machine as it is presented in figure 2 below. After setting such environments, we inspected ICMP, TCP and UDP protocols with different number of packets, size of data and different time interval. We measured throughput and response time when the packet transmission took place by just crossing different nodes to know exactly its performance in the worst-case scenarios. We run the simulation capturing the data using Wireshark for a minimum of twenty seconds in order to examine the effect of path computational time and the time required for negotiation between hosts for example, window size agreement that the receiver can process the bytes of data at a given time interval. During packet transmission, zero TCP window size advertisement is often sent due to various factors including, queue size, link speed and buffer size of the target device, this becomes the cause for packet loss across the entire network, so we enabled the packets to be sent through many links other than the best alternative path through the policies applied inside distributed controllers. Especially, during link failure scenarios, the communicating end points were forced to follow the other links than that they were using before link failure, this may cause significant delay, however, the nearby controller in collaboration with MPLS computes the alternative path in such a way that there would be no significant latency.

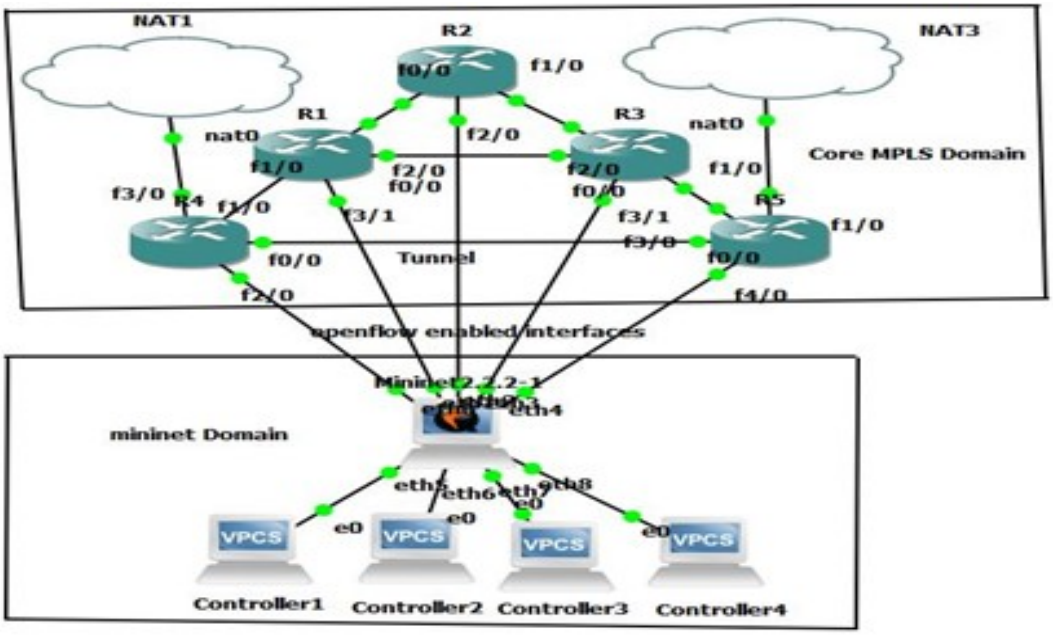

Figure 2: Integrated Topology

4. RESULT AND DISCUSSIONS

4. 1. Throughput

Throughput is a measure of error free packets transferred in a given time interval compared with total number of packets transferred. [14], expressed throughput as a measure of an amount of unit of information a system can process in a given period of time. The integration of MPLS with OpenFlow improved the throughput of the network. We enabled MPLS to operate with SDN controllers through path computation element protocol (PCEP) so that it can handle low-level traffic flows by the help of standalone openvswitches, and the low-level flows are directed to MPLS pipe lines instead of being sent to SDN controllers pipe lines. This approach reduced the load of the controllers during simultaneous transmission of packets among all classes of users, and then packet transmission becomes faster and faster. Not only this, it also reduced the packet loss occurred due to offloading the controllers and queuing delay, hence, it enhances the throughput. Using distributed controllers even let the packet transmission be very much faster than the network with a single central controller which in turn results in an increasing the number of error-free packets transferred and better throughput. We measured the overall throughput of the network when there is simultaneous transmission among all classes of users not solely for each class of users, such as low-level, medium-level and high-level ones. Then we presented our findings in comparison with other works in different scenarios for example, in link failure scenario, size of data transferred and number of hops. The result is illustrated in the following figures.

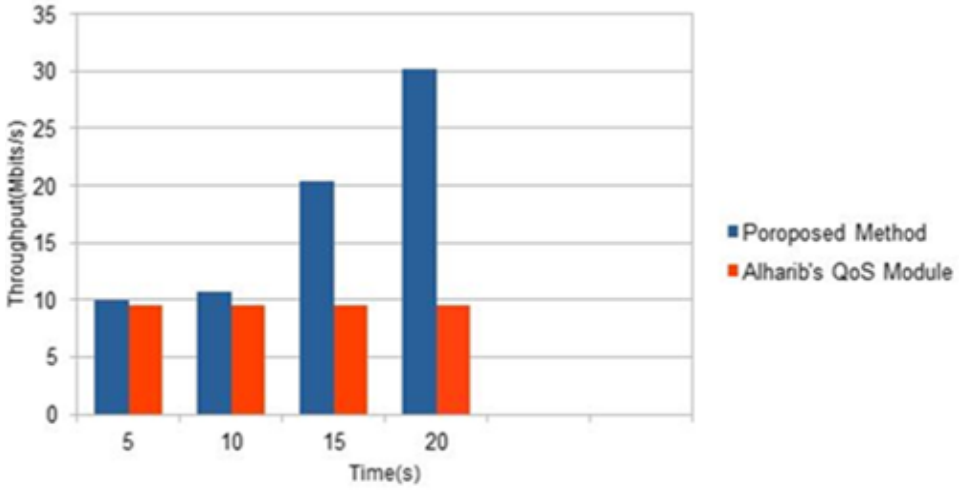

Figure 3: Throughput versus number of Hops

The above figure illustrates that the proposed method is better than Alharib's QoS module [6] in terms of average throughput when packet transmission took place in the same number of hops and in the same time interval. The average throughput that the proposed method achieved is 30.4mbps which is twofold better than Alharib's QoS module. The subsequent graph presents the throughput of proposed method in link failure scenario compared with Yaacob, et al.'s [15] model in link failure scenario. We ran the link failure scenario by shutting down the one link among many links that connect hosts with openvswitches, and letting hosts to communicate one another using other than the shortest path. The main reason for achieving this much throughput is handling packets on both the data plane and the control plane by the help of MPLS and standalone openvswitches that act as bridges, using distributed controllers for balancing the loads and even bandwidth isolation helped us to utilize the dedicated bandwidth effectively and efficiently.

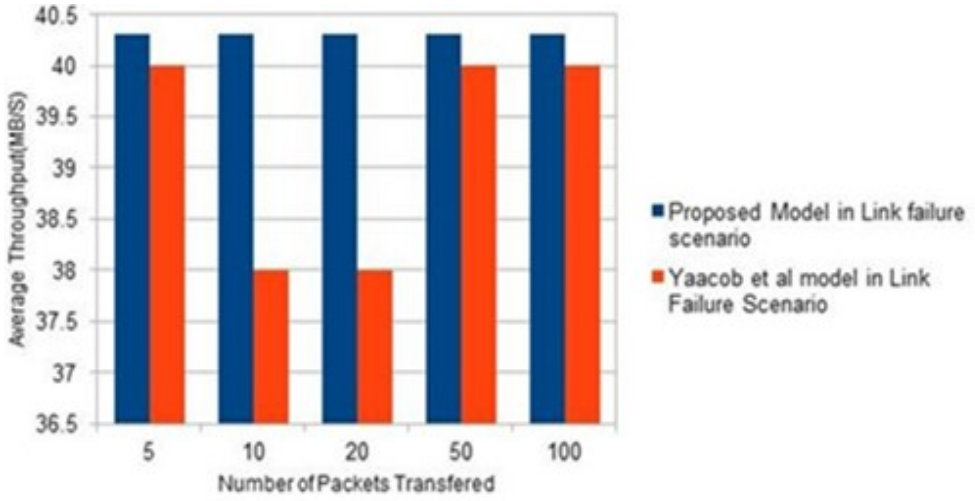

Figure 4: Throughput versus Number of packets

Figure 4 shows our proposed method is much better than Yaacob, et al.'s model [15] in link failure scenario, i.e., the proposed model having 45mbps bandwidth achieved a constant throughput of 40.3mbps within **20ms** time interval, whereas Yaacoob, et al.'s model is often varying, and the maximum throughput of their model is below **40mbps.** The subsequent figure illustrates the throughput against size of data in comparison with NetFPGA open-source router.

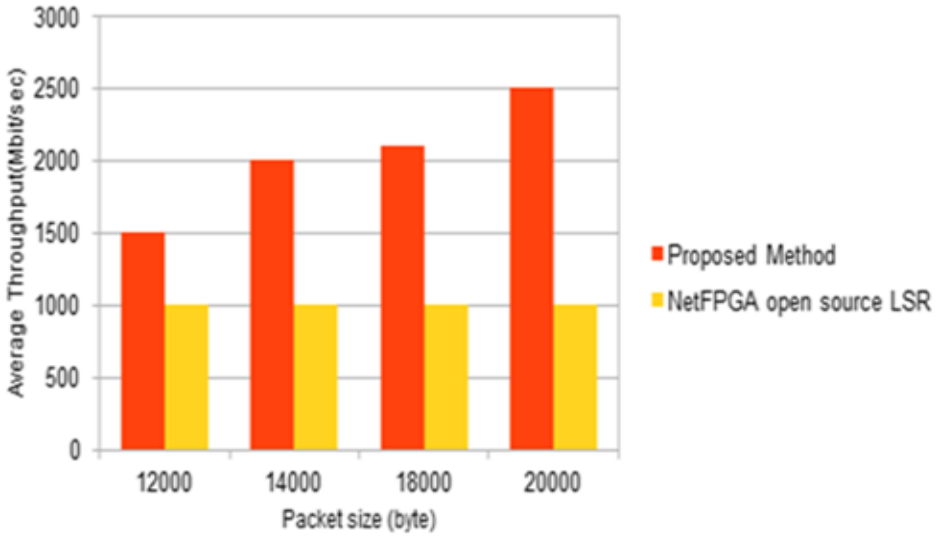

Figure 5: Throughput versus Size of Data

From figure 5, we can conclude that our proposed method is much better than NetFPGA open-source label switching router **[8]** since the throughput of the proposed method having unlimited link capacity increases as the number of byte increases, and the average throughput reaches **2500mbps**. But in case of NetFPGA, the throughput stays constant although the number of byte increases. It is known that the transmission capacity of NetFPGA open-source label switching router is dependent on the network interface card (NIC), but not in case of mininet. The main reason for scoring this much amount of throughput is handling packets on both the data plane and the control plane led the entire packet transmission process to be taken place faster and faster with no significant transmission errors caused by queuing delay and congestion and even using distributed controllers as a load balancing scheme reduced queuing delay, transmission latency and drop of packets caused by congestion and time to live (TTL) expires unlike single central controller.

4. 2. Response Time

The ability of handling packets on both the data plane and the control plane allowed network operations to be quicker and quicker, as a result, there is no more queuing delay. We measured the response time for ten consecutive experiments, and then we have taken the maximum value since it has high discriminant power. In every experiment, the response time is decreased compared with previous works. In our method, low-

level packets are handled on the data plane separately from medium-level and high-level packets, this avoids queuing delay and other transmission delay caused by many loads on the controllers. Distributed controllers also helped us to balance the load across the entire network resulting better transmission speed with admirable response time. We measured the overall response time of the network when there is simultaneous transmission of packets among all class of users that is when there is too much traffic flow not end point level solely for each class of users, such as low-level, medium-level and high-level ones. The following figure illustrates how much the response time is reduced with respect to other methods.

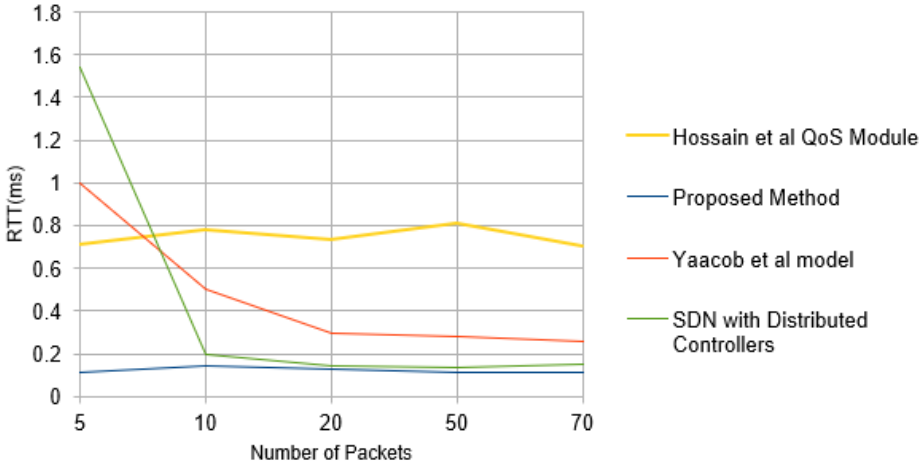

Figure 6: Response time versus Number of Packets

The above figure shows that the proposed method is better than those three methods in RTT; in all cases, the RTT of the proposed method is below **0.2ms**. this is due to the fact that the load on the controllers is reduced by the help of MPLS since it is able to process low-level packet flow by the help of standalone switch that are able to process small group of packet flows called mice flows [1]. Distributed controllers also share the loads, therefore, there would be no more waiting time and queuing delay, even bandwidth isolation helps us to utilize the dedicated bandwidth effectively and efficiently that led the packet transmission to be faster and faster with admirable performance. We compared our method with those methods using packet transmission in the same hop counts and the same number of packets. The figure shows that the RTT decreases as the number of packets increases, this is due to the fact that there is no need of waiting for path establishment or path computation once the path is set for the first group of packets.

4. 2. 1. Effect of Operational Aspects of Switches on Latency

The type of switch matters on the overall performance of software defined network [1]. Since the operating system they use varies, the performance often varies. In our experiments, we used three different types of

switches namely, OVSbr, OVS and user space switches, then we found that there is slight difference interms of minimum, maximum and average response time between them. It is known that OVSbr acts as a bridge, and it is expected to be better in all cases, however, it is not as we expect it to be. Even user space switch is better than OVS in minimum RTT. What matters is the thing we do on the control plane and the way we follow on handling the packets. The effect of the type of switches on response time is presented in the following table.

Table 1: The Impact of the Type of Switch on Response Time

Number of Packets	Type of Switch	Round-Trip Time in(ms)		
		Minimum	Maximum	Average
12	OVS	0.021	3.737	0.380
	OVSbr	0.025	0.147	0.039
	User space switch	0.039	1.611	0.230
20	OVS	0.015	2.286	0.205
	OVSbr	0.025	0.136	0.063
	User space switch	0.032	1.670	0.317
30	OVS	0.013	3.045	0.151
	OVSbr	0.025	0.283	0.067
	User space switch	0.037	1.384	0.291
50	OVS	0.020	7.847	0.223
	OVSbr	0.023	0.269	0.056
	User space switch	0.048	1.541	0.477
100	OVS	0.016	18.152	0.246
	OVSbr	0.022	0.303	0.055
	User space switch	0.333	2.688	0.338

4. 3. Discussion

SDN is an emerging network paradigm that ensures open networking with the needs of many investigations regarding its performance to enable it supports as many numbers of users as possible. In this paper, we investigated throughput and response time of SDN using distributed controllers, MPLS-based data plane and policy-based routing with respect to different scenarios i.e., number of hops, number of packets and size of data transferred. When we study throughput, we were able to use the entire bandwidth that mininet supports and beyond using bandwidth isolation, then we ran three different scenarios namely throughput

against number of packets, throughput versus size of data and throughput versus number of hops. From our findings, the maximum throughput that the proposed system reached are **30.4mbps, 40.3mbps and 2500mbps** for throughput versus number of hops, throughput against number of packets and throughput versus size of data scenarios respectively. Regarding response time, we ran ten consecutive experiments, then we took the maximum result obtained considering the worst-case scenario. From ten consecutive experiments, we got **0.2ms** of response time; that is the maximum response time recorded when there is maximum link utilization in the worst-case scenario. When we investigate response time, we worked with three different types of switches namely, OVSbr, OVS and user space switches, then we got slight difference in terms of response time; the maximum response time taken is recorded with OVS switch that is **18.152ms**.

5. CONCLUSION AND FUTURE WORKS

5. 1. Conclusion

In this paper, we tried to investigate the response time and throughput of SDN and the impact of the type of switch on the performance of SDN. In our study, we have used, distributed controllers with MPLS and policy-based routing together, then we measured parameters, such as response time and throughput using different scenarios. We tried to handle packet processing on both the data plane and the control plane. We used mininet with GNS3 for simulation and we used open-source controllers, such as POX, ONOS and opendaylight. While conducting experiments, we didn't observe significant difference between controllers although there was slight difference occasionally. From our findings, we concluded that the integration of MPLS on the data plane is an efficient method to enhance the overall performance of SDN.

5. 2. Future Works

From our findings, optimizing the performance of different types of individual openvswitches will be extended as a future work.

Acknowledgement

We would like to acknowledge all of our colleagues and friends around us for their valuable support to accomplish this work.

References

[1] F. Xenofon , M. . K. Mahesh and K. Kimon , "Software Defined Networking Concepts," in *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*, John Wiley & Sons, Ltd, 2015, pp. 21-44.

[2] D. Bhat, K. Rao and L. N R, "A Survey On Software-Defined Networking Concepts And Architecture," *International Journal of Science, Technology & Management,* pp. 123-141, 2015.

[3] R. Khalili, Z. Despotovic and A. Hecker, "Flow setup latency in SDN networks," *IEEE Journal on Selected Areas in Communications,* pp. 1-9, 2018.

[4] N. A. Burno, M. Marc, N. N. Xuan , O. Katia and T. Thierry , "A Survey of Software-Defined Networking: Past,Present, and Future of Programmable Networks," *IEEE,* pp. 1617-1634, 2014.

[5] S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou and D. Simeonidou, "Integrated Openflow–GMPLS control plane: an overlay model for software defined packet over optical networks," *Optics Express,* pp. B421-B428, 2011.

[6] A. Faisal , "SDN-Based Mechanisms For Provisioning Quality Of Service To Selected Network Flows," Lexington, 2018.

[7] A. M. Hossain, M. N. Amin Sheikh, S. M. Shawon , S. Biswas And M. A. I. Arman, "Enhancing And Measuring The Performance In Software Defined Networking," *International Journal Of Computer Networks & CommunicationS (IJCNC),* VOL. 10, NO. 5, PP. 27-40, 2018.

[8] J. Kempf, S. Whyte, J. Ellithorpe, P. Kazemian, M. Haitjema, N. Beheshti, S. Stuart and H. Green, "OpenFlow MPLS and the Open Source Label Switched routers," in *23rd International Teletraffic Congress*, 2011.

[9] A. Tootoonchian, S. Gorbunov and Y. Ganjali, "On Controller Performance in Software-Defined Networks Topics in Management of Internet, Cloud, and Enterprise Networks and Services.," in *2nd USENIX conference on Hot*, San Jose, 2012.

[10] T. M. Mohammad , A. Behzad , M. Nader and C. Luca , "SDN-Based Resource Allocation in MPLS Networks: A Hybrid Approach," in *concurrency and computation practice and experience*, 2018 John Wiley & Sons, Ltd., 2018, pp. 1-11.

[11] J. Bellessa, Implementing MPLS With Label Switching In Software-Defined Networks, Urbana, 2015.

[12] Y. Sinha, S. Bhatia, V. S. Shekhawat and G. S. S. Chalapathi, "MPLS based Hybridization in SDN," in *2017 Fourth International Conference on Software Defined Systems (SDS)*, India, 2017.

[13] K. Kaur, J. Singh and N. S. Ghumman, "Mininet as Software Defined Networking Testing Platform," in *nternational Conference on Communication, Computing & Systems (ICCCS–2014*, 2014.

[14] K. Alemayehu, "Analyzing Impact of Segment Routing MPLS on QoS," Addis Ababa, 2019.

[15] Y. N , A. A , A. R B , W. M N M , I. Z and S. Miam , "Performance analysis of Software Defined Network (SDN) in link failure scenario," in *IOP Conference Series: Materials Science and Engineering*, Bogor, 2019.