Performance Analysis of POX, Open vSwitch and Open Day Light SDN Controllers on Cloud

Neeraj Chauhan, Manu Sood

Abstract-* Software Defined Networking (SDN), as a revolutionary technology is changing the way traditional computer networks used to function. It is a replacement of traditional networks where control of the network was embedded with the data layer. Software Defined Networks separates control layer from data layer which enables network administrators to reconfigure the network easily without making any physical change in the network. An SDN network has various components like switch, controllers and hosts. Generally, SDN controllers are considered the "brain" of this network. In this paper, firstly, performances of three SDN controllers POX, Open vSwitch (OVS) and OpenDayLight (ODL) are analyzed for single topology on cloud on the basis of three parameters Round-Trip-Time, Average Time and Total Time. Results show that OVS controller outperforms POX and ODL controllers. Later, the performance of these three controllers has also been analyzed for different topologies on local machines.

***Index Terms:* Google Cloud, Open vSwitch (OVS) controller, OpenDayLight (ODL) controller, POX controller, SDN controller, Software Defined Networking (SDN).**

I. INTRODUCTION

SDN is the latest and promising paradigm in computer networks. It separates data plane from the plane that controls the networking tasks. Data plane is the layer in the networking architecture that handles the flow of data whereas on the other hand, control plane is another layer where the SDN controller is housed. Thus, any network administrator is able to manage overall network effectively through computer programs and commands as per the required dynamic policies. In SDN, Switches placed in data plane simply act as a packet forwarding devices and the entire network is controlled by a logically centralized software program which resides in control plane [1]. SDN architecture makes things such as modification of the configuration of the network devices (routers, firewalls and switches) very convenient which otherwise is very complicated task in traditional networks [2]. There are some significant inherent limitations in traditional networks such as complex management, vendor dependent network and predefined policies of forwarding devices. SDN eliminates these limitations by having programmable networks and separating control and data plane, which forwards the network traffic as per decisions made by controllers. There are many definitions of SDN, a clear and suitable definition of SDN is given as: "In the SDN architecture, the control plane and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications" [3].

In an SDN architecture, the infrastructure devices like switches and routers have been designed to act like engines which process the incoming data packets so as to forwards these packets towards their respective destinations. The forwarding of these packets is based upon the logic based set of rules programmed into the controller(s) belonging to the control plane. Typically, this programmed controller is designed to be executed on some remote server so that with the help of a set of certain standard command, it can facilitate the forwarding elements to establish secure connection for the purpose of communication. Primarily, the functional architecture of SDN is mainly based on three layers as shown in Fig. 1.

The main constituents of the infrastructure Layer (also known by the name of data plane) are switches which can either be physical or virtual. These switches can be accessed through an open interface known as Southbound API and are responsible for forwarding and switching of the packets during network data flow.

The second one, the Control Layer (known by the name of control plane too) is characterized by a set of open APIs based programmed controllers. They have in-built programmed functionality that helps in supervising the behavior of forwarding policies implemented in the controllers of SDNs. Application layer is the third layer of this architecture as shown in Fig. 1. The Control Layer lies embedded between this layer and the Infrastructure Layer. In fact, the controllers of the Control Layer interact/communicate with the other two layers with the help of three different sets of APIs. The Northbound API interfaces the Control Layer with Application Layer and provides a common application development platform as software API. The Southbound API acts as the interface between Control Layer and Infrastructure Layer with an additional focus is on defining protocol(s) for communication among the elements of control plane and the forwarding devices. There are various data and control layer communication protocols used for interaction among

Revised Manuscript Received on May 24, 2019.

Neeraj Chauhan, Department of Computer Science, Himachal Pradesh University, Shimla, India.
Manu Sood, Department of Computer Science, Himachal Pradesh University, Shimla, India.

elements of these planes, namely OpenFlow, Opflex, Yang and Net Conf protocols.

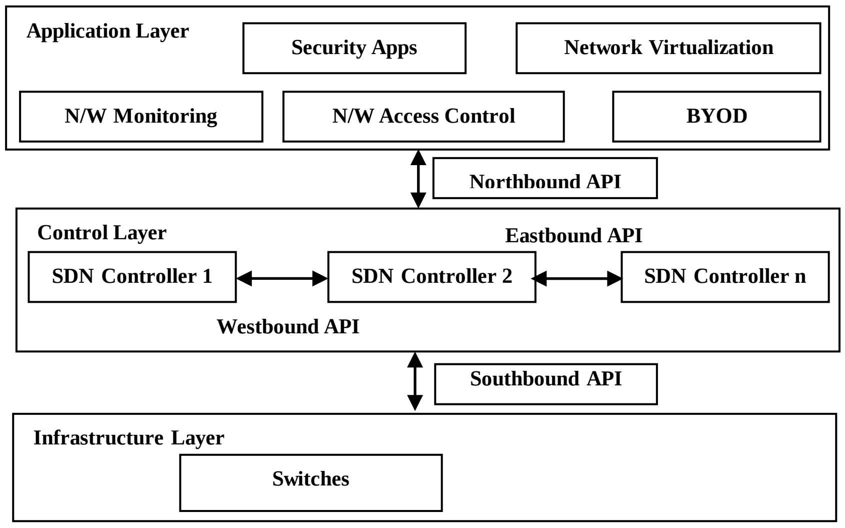

Fig. 1. Architecture of SDN

The Eastbound and Westbound APIs provide the interfaces among the group of controllers in the Control Layer and these ensure the availability and scalability in the SDN. The Application Layer supports the applications based on the business logic of the end users through the network services of the SDN e.g. security component of the applications, visualization of network components, communication support etc.

Fig. 2 exhibits the functioning of the SDN [4]. It shows the complete communication process in five steps initiated by the sender and terminating at the receiver as shown below.

Step 1: The first packet of a new communication session between sender and receiver arrives at the switch from the sender.

Step 2: The switch initiates the procedure for finding a flow entry rule for this packet in its data flow table(s) stored in the cache memory. If a corresponding match is found, then Step 3 and 4 are skipped and the instructions related to such specific entry are processed and Step 5 is performed.

If no corresponding match related to the incoming packet is found in the flow table, then:

Step 3: The packet will be forwarded to the controller over a highly secured channel using a communication protocol (e.g. OpenFlow, ForCES, PCEP etc.) from the southbound API.

Step 4: The Southbound API like OpenFlow, ForCES, PCEP initiates the forwarding of the packet to the SDN controller using a secure communication channel for either adding or deleting or updating the flow table entries in the corresponding flow table(s). Both reactive and proactive policies can be used for these operations. The routing algorithm for the onward transmission of the packet too gets executed.

Step 5: The packet is forwarded to the next switch in the flow path or to an appropriate port corresponding to the receiver.

This paper consists of seven sections which are organized in a way that Section II explains a brief architecture of the SDN and its APIs. This section also explains how SDN work means how SDN perform its basic task of packet forwarding when there is no flow entry in at the switch level in the data plane. Section III gives a detailed literature review that has been carried out during this study. Section IV explains research methodology adopted. This section mentions various steps carried out to perform the analysis of the three controllers POX, OVS and OpenDayLight. Section V describes results and analysis of the experimental work. Section VI concludes the present research work and Section VII discusses the future scope of the study.

II. SDN CONTROLLERS

SDN controller is called as the 'Brain of SDN' because it manages and configure the overall network. A controller resides in Control layer (Servers) of the SDN architecture and provides various services to data layer and application layers with the help of various southbound and northbound APIs as discussed in Section I of this paper. SDN supports various controllers provided by various vendors [5]. Controllers are programmed by developers in different programming languages like C, C++, java, Python and Ruby. SDN controllers are based on protocols such as OpenFlow, discussed later in section IV. The major functions of the controllers are listed here as below:

- The controller configures and manages the complete functionality of the network
- Management of flow control for dynamic networking
- Optimization of network path for onward application traffic
- Installation of new flow entries on switching devices

The controller is the main component of the SDN. This research work presents a detailed study of three SDN controllers; POX, OVS, and OpenDayLight used in SDN. Some important characteristics of these three controllers have been discussed here. A) POX is an SDN controller programmed in Python and it is similar to NOX controller.

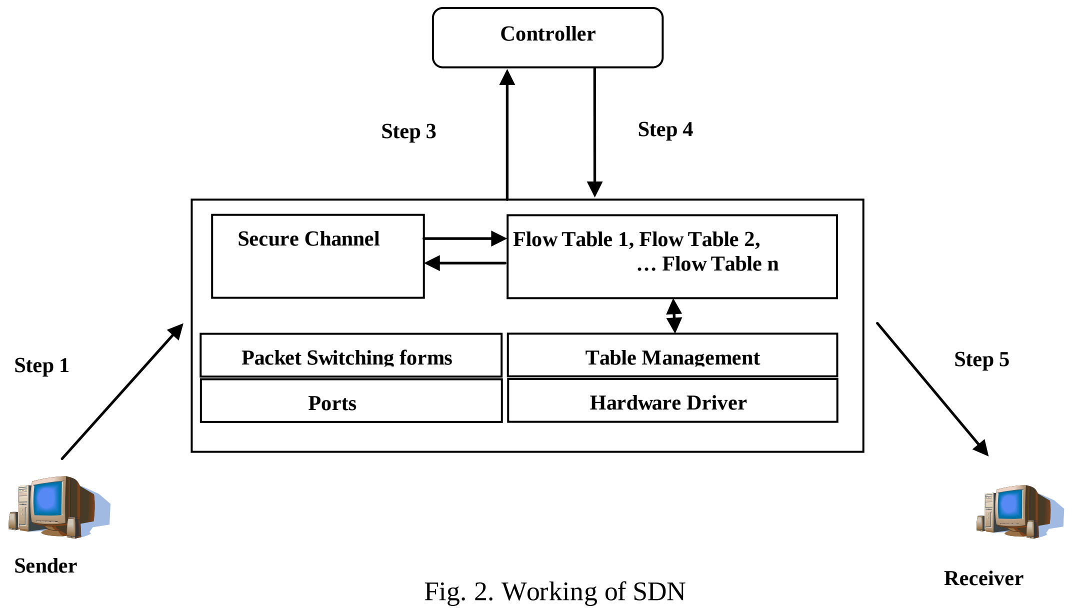

Fig. 2. Working of SDN

The OpenFlow version supported by POX is OpenFlow 1.0. POX is an open source and also supports Graphical User Interface. B) Open vSwitch (OVS), another simple OpenFlow protocol based controller has been designed to manage as many number of switches as required. These switches can be configured to perform the functions of either L-2 MAC-learning switches or hubs. It comes preinstalled with MININET that we have installed on Linux VM on Google Cloud and can be used for preliminary testing of networks supporting OpenFlow. C) OpenDayLight (ODL) is another open source software project. A JVM container has been used for the implementation of its SDN controller [6]. Any platform that supports Java can be used for its deployment. Its architecture supports the OSGi framework [7]. Many mainstream organizations of the fame of IBM, Cisco, Ericsson, NEC etc. are continuously supporting its development and implementation. Major releases of ODL are Hydrogen (February 2014), Helium (September 2014), Lithium (August 2015), Berillium (March 2016), Boron (December 2016), Carbon (May 26, 2017), Nitrogen (May 11, 2018), Oxygen (March 22, 2018), Fluorine (August 30, 2018).

III. LITERATURE REVIEW

This section gives a detailed literature review on SDN. It includes major findings and research work carried out by various researchers. During this study various research papers, survey papers, books and online articles have been read. This section gives an overview of the literature that has been studied during the present study.

Raja et al. in [8] compare and contrast two main strategies used for the implementation of SDN: a) proprietary and b) open source and non-proprietary where their adoptability by various researchers and/or organizations has been found to be based on their respective operating principles, salient features, limitations, strengths and weaknesses. The authors opine that though the non-proprietary implementation may have a cost component attached to them but are more stable and properly backed up with necessary support. The open source implementations may be almost without costs and help in fast tracking the implementation of SDNs but suffer from certain issues. They recommend the hybrid approach for the better proliferation of SDNs.

Kim and Feamster in [9] propose the design and implementation of Procera, an event-driven network framework built on the concepts of SDN. This Procera simplifies various aspects of network operations and its management and promises options for specifying richer network policies. OpenFlow protocol has been used for communication among the Procera controller and the switches. They draw their rich experiences from its deployment in both a home and campus network to show that Procera also reduces the complexity of network management.

Rahamatullahet al. in [10] propose a method called Multi-Criteria Decision Making, known as Analytic Hierarchy Process (AHP) to choose the best open source SDN controller. After studying and analyzing various SDN controllers available, they choose the five topmost controllers. This selection is based on the characteristics like available interfaces, modularity, productivity on one hand and on their current deployment and utilization on the other. They conclude that among these five top performing controllers, the best controller is ‘Ryu”.

Kuretz et al. in [11] present an exhaustive account of Software Defined Networking in their paper. The paper starts from its motivation, roots, comparison with traditional networking, standardization initiatives, its building blocks; continues to present the analysis of APIs, hardware components, network virtualization layer, operating systems, programming languages, and applications. They also highlight some of the cross layer interfacing problems, other major challenges, a snapshot of the ongoing research on this topic and latest available opportunities.

Paul et al. in [12] depicts a brief history of networking followed by a discussion on the significant networking technologies and how OpenFlow as well as SDN have been created.

Nadeau and gray in [13] highlight the definitions, protocols, and standards for the programmable SDNs. The authors also present detailed discussions on linkages among big data, datacenters, network virtualization, bandwidth manipulations and some other components of SDNs with the help of some interesting use cases.

Braun and Menth in [14] explain the operation of OpenFlow along with its features for versions ranging from 1.0 to 1.4 followed by a discussion on possible choices for the architectural design for SDN and their performance. They propose that the controller may be centralized either physically or logically. They summarize their paper by explaining the advantages/disadvantages of these approaches along with challenges in SDN built upon OpenFlow.

Zinner et al. in [15] explain SDN, its interfaces with the help of salient features.

Stancu et al. in [16] present a comparison of POX, RYU, ONOS and OpenDayLight SDN controllers by simulating these four controllers in Mininet environment with one controller, 15 switches and 16 hosts using the tree topology. Based upon their experimentation, the authors show that in comparison to ONOS and OpenDayLight, POX is the easiest to run. They also highlight that development in this environment is comparatively easier to learn.

Priyadarshini et al. in [17] discuss about how performance metrics of the SDN controllers based upon simulation results can be measured and analyzed. They also explain the working of various controllers and stress upon the performance enhancement as one of the crucial challenges for the SDN architecture and its security. Another major finding is that the load and performance are inversely proportional to each other under high data traffic conditions.

Yamei et al. in [18] analyze the implementation architecture of two open source SDN controllers, ODL and ONOS on four parameters using simulations in an environment consisting of IXIA test instruments, Cbench and Mininet. The authors finally show that the performance of ONOS is better on GUI, clusters, link-up, switch-up and throughput; while that of ODL is better in terms of topology discovery and stability.

IV. RESEARCH METHODOLOGY ADOPTED

We have used Mininet emulator to compare the performances of POX, OVS and OpenDayLight SDN controllers on cloud for single topology in the first phase and for three other topologies (linear, reversed and tree) on the local machine in the later phase. In the first phase, virtual machines have been created on Google Cloud Platform (GCP) and in the second phase, virtual machines have been created on a local machine. All the necessary tools and SDN controllers have been installed. After installing all the controllers, results have been obtained for Round-Trip-Time (RTT), Average Round-Trip-Time (AT) and Total Time (TT) using PING command. As a thumb rule, low RTT leads to low latency. Network Latency is “an expression of how much time it takes for a packet of data to travel from designated sending endpoint to the designated receiving endpoint [19]. Round-Trip-Time (RTT) also known as Round-Trip-Delay “is the time required for a signal pulse or packet to travel from a specific source to a specific destination and back again” [20].

Although these two terms are almost similar but the delay at the echoing endpoint is also included RTT. The only topology we have created on cloud is “single topology” as shown in Fig. 3.

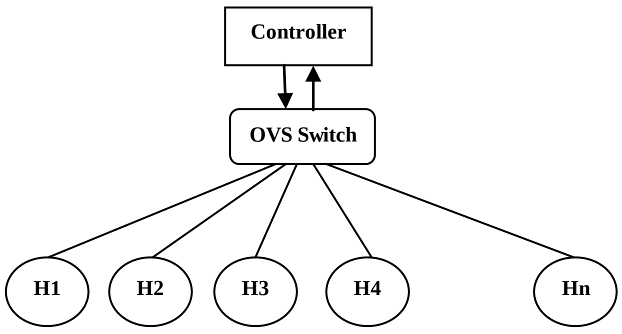

Fig. 3. Single Topology

In Fig. 3, H1, H2, H3, … Hn are the virtual hosts with OVS Switch, an open-source, OpenFlow switch that works as a virtual switch in the virtualized environments. It is used as multilayer software for interconnecting virtual devices in the same host or among different hosts [21] [22].

All the three controllers used by us work on OpenFlow protocol for various topologies on local machines and on cloud where these topologies specify the messages and their formats in the overall SDN environment. These also prescribe the reaction of various devices under various scenarios and their response to incoming commands from the controller. Various available versions of OpenFlow protocol till date are 1.0, 1.1, 1.2, 1.3, 1.3.1, 1.3.2, 1.3.3, 1.3.4, 1.3.5, 1.4.0, 1.4.1, 1.5.0, 1.5.1 [23]. More details on the history of this protocol can be found in [12].

In this study, the effect of the above mentioned three controllers for single topology on cloud and different topologies on local machines on the parameters RTT, TT and AT have been analyzed for two cases: a) While increasing number of hosts in the given topology from 5 to 25 in steps of 5, and b) While increasing number of packets being transmitted from 25 to 100 in the steps of 25. The steps involved in our performance analysis of the three SDN controllers are given below:

1. First create a new GCP project and enter the billing details [24]
2. Create virtual machine Instance on server [25]
3. Download and Install Putty [26]
4. Download and Install Mininet [27]

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

5. Download and Install POX controller [28]
6. Download and Install OpenDayLight controller [29]
7. Download and Install OVS controller (pre-installed on Mininet)
8. Analyze comparative results for three SDN controllers for different parameters on cloud for single topology and on a local machine for linear, reversed and tree topology.

V. EXPERIMENTAL RESULTS AND ANALYSIS

The results for all the three SDN controllers (POX, OVS and OpenDayLight) on a virtual machine installed for single topology on GCP under two cases, Case 1 and Case 2 have been presented here for Round-Trip-Time, Average Round-Trip-Time and Total Time in table I and table II followed by the corresponding bar charts. In case 3, the performances of these three controllers have been measured on a local machine for three different topologies.

CASE 1. NUMBERS OF HOSTS ON CLOUD FOR SINGLE TOPOLOGY ARE INCREASED (NO. OF PACKETS = 50)

Table I. Performance of SDN Controllers with Increasing Numbers of Hosts for Single Topology on Cloud

Resulting Parameter (in ms)	No. of Hosts	POX	OVS	ODL
Round-Trip-Time (RTT)	5	10.7	3.81	13.9
	10	22.3	6.95	14.9
	15	48.6	4.94	14.4
	20	49.4	4.16	15.2
	25	36.1	4.32	17.3
Average Round-Trip-Time (AT)	5	9196	9170	9176
	10	10198	9201	9178
	15	10201	9192	9181
	20	10202	9172	9170
	25	11239	9178	9172
Total Time (TT)	5	1.142	0.447	1.461
	10	2.097	0.776	1.571
	15	4.516	0.559	1.517
	20	4.560	0.482	1.599
	25	3.094	0.505	1.814

- RTT for OVS controller is the least and for POX, the highest as shown in Fig. 4 (RTT in ms on y-axis, No. of hosts on x-axis).
- AT for ODL as well as OVS is the least (almost equal for both) and for POX, again the highest as shown in Fig. 5 (AT in ms on y-axis, No. of hosts on x-axis).
- TT of OVS controller is the least and for POX, the highest as shown in Fig. 6 (TT in ms on y-axis, No. of hosts on x-axis).

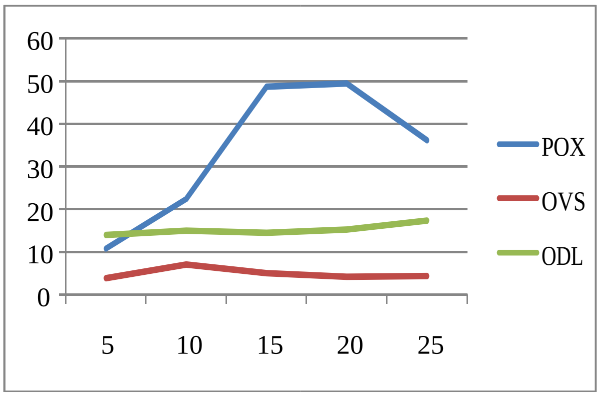

Fig. 4. RTT for POX, OVS and ODL

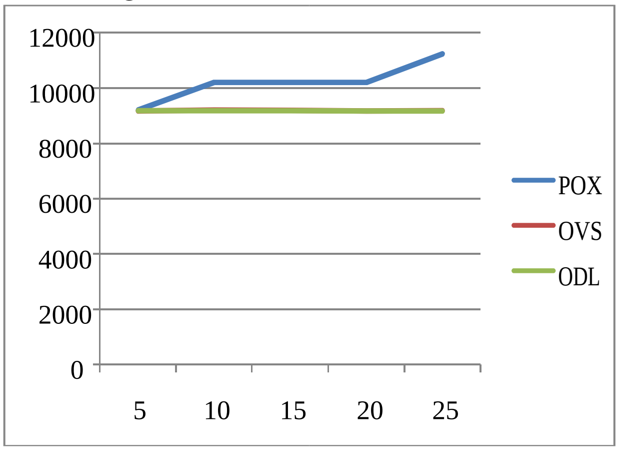

Fig. 5. AT for POX, OVS and ODL

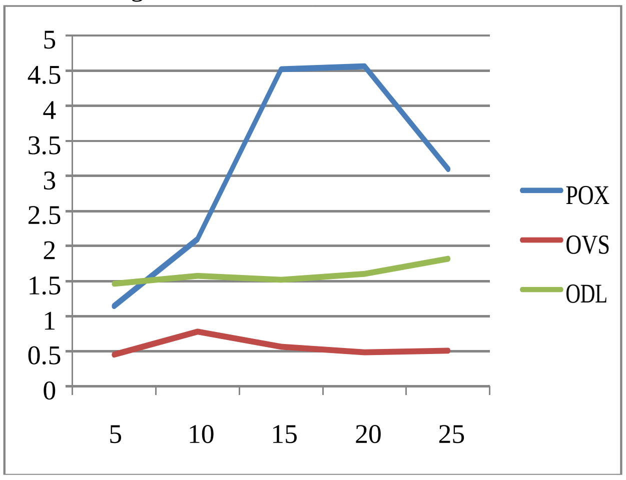

Fig. 6. TT for POX, OVS and ODL

CASE 2. NUMBERS OF PACKETS ON CLOUD FOR SINGLE TOPOLOGY ARE INCREASED (NO. OF NODES = 15)

- AT for all the three controllers increases with the increase in the number of packets and is almost same for all the three controllers as shown is Fig. 7 (AT in ms on y-axis, No. of packets on x-axis).
- TT for POX controller is the highest and for OVS, the lowest

although similar to ODL as shown in Fig. 8 (TT in ms on y-axis, No. of packets on x-axis).

Table II. Performance of SDN Controllers with Increasing Numbers of Packets for Single Topology on Cloud

Resulting Parameter (in ms)	No. of packets	POX	OVS	ODL
Average Time (AT)	25 50 75 100	24545 50120 75686 101246	24546 50176 75744 101369	24557 50182 75765 101323
Total Time (TT)	25 50 75 100	4.792 1.034 1.096 1.160	0.208 0.057 0.073 0.049	0.648 0.052 0.067 0.178

Note: We have not considered RTT here because it would have same results as in the case 1.

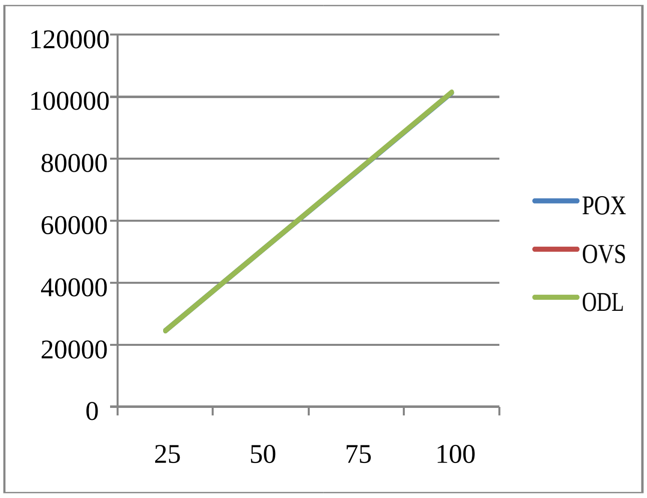

Fig. 7. AT for POX, OVS and ODL

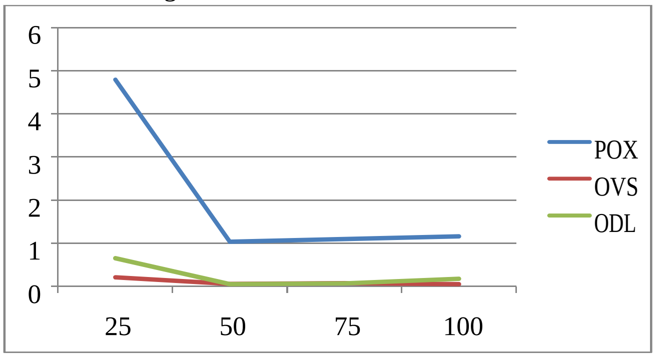

Fig. 8. TT for POX, OVS and ODL

All the results from above graphs reflect that the OVS controller is the best among all the three controllers because it has low RTT and TT as compared to other two SDN Controllers.

CASE 3. NUMBERS OF HOSTS ON LOCAL MACHINE ARE INCREASED FOR LINEAR, REVERSED & TREE TOPOLOGIES (NO. OF PACKETS = 50)

Like single topology that has been implemented on GCP, we have implemented three other topologies (linear, reversed and tree) without GCP on the local machines measuring RTT, AT and TT for all three controllers (POX, OVS and OpenDayLight). The values of parameters measured have been shown in tables III, IV & V.

Table III. Performance of SDN Controllers with Increasing Numbers of Hosts for Linear Topology on Local Machine

Resulting Parameter (in ms)	No. of Hosts	POX	OVS	ODL
Round-Trip-Time (RTT)	5 10	12.7 24.4	5.0 8.75	15.9 16.7
Average Round-Trip-Time (AT)	5 10	9396 10398	9370 9401	9376 9378
Total Time (TT)	5 10	2.142 3.097	0.756 0.986	2.346 2.443

Table IV. Performance of SDN Controllers with Increasing Numbers of Hosts for Reversed Topology on Local Machine

Resulting Parameter (in ms)	No. of Hosts	POX	OVS	ODL
Round-Trip-Time (RTT)	5 10	14.7 26.4	7.04 10.68	17.09 18.07
Average Round-Trip-Time (AT)	5 10	9596 10598	9567 9607	9566 9783
Total Time (TT)	5 10	3.01 4.31	1.01 1.32	2.7 2.9

Table V. Performance of SDN Controllers with Increasing Numbers of Hosts for Tree Topology on Local Machine

Resulting Parameter (ms)	No. of Hosts	POX	OVS	ODL
Round-Trip-Time (RTT)	5 10	15.47 28.32	9.12 12.86	19.56 18.47
Average Round-Trip-Time (AT)	5 10	9753 10765	9750 9801	9776 9958
Total Time (TT)	5 10	3.5 4.86	2.1 1.7	2.9 3.2

It can be observed from tables III, IV and V that for linear, tree and reversed topologies, RTT and TT are the least and but the best for OVS controller. As far as AT is concerned, POX is the poorest performer in all three topologies.

VI. CONCLUSION

For the upcoming generation of networks, Software Defined Networking using OpenFlow protocol are considered to be the most deployed networking architecture. OpenFlow protocols provide high standards for routing and delivery of packets on a switch in a network. In this paper, three OpenFlow based controllers have been discussed and implemented to analyze their performance on RTT, AT and TT. The numbers of hosts and the numbers of packets transmitted are varied separately under cloud environment for single topology only in the first phase and number of hosts varied for linear, reverse and tree topologies under local environment. All the evaluations for the first phase have been carried out using Mininet Emulator which has been installed and configured on Google Cloud and for the second phase, these have been carried out on a local machine. This paper presents the comparison of performance of these three SDN controllers; POX, OVS and OpenDayLight on above mentioned three parameters, conditions and topologies. The results show that OVS controller outperforms the other two SDN controllers on the cloud as far as all the three parameters are concerned. It has also been shown that even when local machine is used for the implementation of these controllers and networks, OVS controller has a definite edge over other controllers.

VII. FUTURE SCOPE

The authors intend to include in future, the study of and working on various protocols (other than overflow protocol) used in SDN and to compare some more SDN controllers such as ONOS, FloodLight, RYU and other SDN controllers. Also, several other topologies for controllers under SDN environment having more complex scenarios are required to be explored and the results have to be evaluated for some more parameter such as throughput and latency which are too suitable performance indicators to measure performance of a SDN.

REFERENCES

1. K. Hyojoon, and N. Feamster. "Improving Network Management with Software Defined Networking", IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.
2. C. Pin-Jui, and Y.Chen. "Implementation of SDN based Network Intrusion Detection and Prevention System", In proceedings of International Carnahan Conference on Security Technology (ICCST), pp. 141146, IEEE, 2015.
3. “Software-Defined Networking: The New Norm for Networks”, Open Networking Foundation, White Paper, Available online at https://www.opennetworking.org/?s=Software-Defined+Networking%3A+ The+New +Norm+for+Networks, Last accessed on 03/02/2019.
4. S. Sakir, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. "Are We Ready for SDN? Implementation Challenges for Software Defined Networks", IEEE Communications Magazine, vol. 51, no. 7, pp. 36-43, 2013.
5. B. N. Astuto A., M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software Defined Networking: Past, Present, and Future of Programmable Networks", IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, 2014.
6. “OpenDayLight”, Available online at http://www.opendaylight.org/ project/technical-overview, Last accessed on 03/02/2019.
7. Alliance, O., OSGi Core Release 5, Specification, March 2012, Available online at https://osgi.org/download/r5/osgi.core-5.0.0.pdf, Last accessed on 03/02/2019.
8. M.H. Raza, C.S. Sivakumar, N. Ali, and B. Robertson. "A Comparison of Software Defined Network (SDN) Implementation Strategies", In proceedings of 2nd International Workshop on Survivable and Robust Optical Networks, Procedia Computer Science, Elsevier ScienceDirect, vol. 32, pp. 1050-1055, 2014.
9. H. Kim, and N. Feamster. "Improving Network Management with Software Defined Networking", IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.
10. K. Rahamatullah, A. Zaalouk, R. Marx, and K. Bayarou. "Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers", In proceedings of World congress on Computer Applications and Information Systems (WCCAIS), pp. 1-7. IEEE, 2014.
11. D. Kreutz, F.M.V. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig. "Software Defined Networking: A Comprehensive Survey", In proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2015.
12. G. Paul, C. Black, and T. Culver, Software Defined Networks: A Comprehensive Approach, Morgan Kaufmann, 2016.
13. T.D. Nadeau, and K. Gray, SDN – Software Defined Networks: An Authoritative Review of Network Programmability Technologies. " O'Reilly Media, Inc.", USA, 2013.
14. W. Braun, and M. Menth, "Software-Defined Networking using OpenFlow: Protocols, Applications and Architectural Design Choices", Future Internet, vol. 6, pp. 302-336, 2014.
15. T. Zinner, M. Jarschel, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "A Compass Through SDN Networks", Report No. 488, Dec. 2013, Available on https://pdfs.semanticscholar.org/8143/ 693ff747e60a639ffaea15697c4b263f1e04.pdf, Last accessed on 03/02/2019.
16. A.L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E.C. Popovici, "A Comparison Between Several Software Defined Networking Controllers", In proceedings of 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), pp. 223-226, IEEE, 2015.
17. M. Priyadarsini, P. Bera, and R. Bhampal, "Performance Analysis of Software Defined Network Controller Architecture – A simulation Based Survey", In proceedings of International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 1929-1935, IEEE, 2018.
18. F. Yamei, L. Qing, and H. Qi, "Research and Comparative Analysis of Performance Test on SDN Controller", In proceedings of First IEEE International Conference on Computer Communication and the Internet (ICCCI), pp. 207-210. IEEE, 2016.
19. Available on https://whatis.techtarget.com/definition/latency, Last accessed on 03/02/2019.
20. Available on https://searchnetworking.techtarget.com/definition/round-trip-time, Last accessed on 03/02/2019.
21. Available on http://www.fiber-optic-transceiver-module.com/ openvswitch-vs-openflow-what-arethey-whats-their-relationship.html, Last accessed on 03/02/2019.
22. Available on https://www.sdxcentral.com/cloud/open-source/ definitions/what-is-open-vswitch/, Last accessed on 03/02/2019.
23. Available on https://www.opennetworking.org/wp-content/uploads/ 2014/10/openflow-switchv1.5.1.pdf, Last accessed on 03/02/2019.
24. Available on https://cloud.google.com/billing/docs/how-to/manage-billing-account, Last accessed on 03/02/2019.
25. Available on https://console.cloud.google.com/compute/ instances?pli=1, Last accessed on 03/02/2019.
26. Available on https://www.putty.org/, Last accessed on 03/02/2019.
27. Available on http://mininet.org/download/, Last accessed on 03/02/2019.
28. Available on https://noxrepo.github.io/pox-doc/html/, Last accessed on 03/02/2019.
29. Available on https://docs.opendaylight.org/en/stable-boron/getting-startedguide/installing_opendaylight.html, Last accessed on 03/02/2019

AUTHORS PROFILE

Neeraj Kumar Chauhan holds an engineering degree in Electronics and Communications from Punjab Technical University, India. Currently he is pursuing M. Tech. in Computer Science from the Department of Computer Science, Himachal Pradesh University, Shimla, India.

Dr. Manu Sood is a Professor in the Department of Computer Science, HPU, India. He had earlier held the additional charge of the Director, University Institute of Information Technology, Himachal Pradesh University, Shimla, India also for four and a half years. He had remained the Chairman of Department of Computer Science, Himachal Pradesh University, Shimla, India too for two terms of two years each. He is an Engineering graduate, has an M. Tech. degree in Information Systems from University of Delhi (DU) with Gold Medal. He also holds the degree of Ph.D. from the Faculty of Technology, DU, Delhi, India. He possesses around 5 years of Industry experience and more than 26 years of academics and diverse administrative experience. His areas of interest in research are Software Engineering, e-Learning, security in WANETs and SDNs.