International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9, Issue-7, May 2020

Achieving Enhanced Network Performance in UDP and TCP Traffic of Software Defined Networking by Selecting Java Based Controllers Decisively

Pinkey Chauhan, Mithilesh Atulkar

Abstract: ***Separating out control plane from the forwarding plane is the core idea of Software Defined Networking (SDN). Control plane has the complete decision making logic and the forwarding plane is restricted to just follow the instructions given by control panel i.e. it is concerned with just forwarding the data. This part of decision making in the control plane is looked after by the Controller, so it is the most important component of the SDN. Depending upon the main objective of the created network a wise selection of the controller is important so that it can provide maximum performance for the said objective. Depending upon the need, if a right controller is selected it will give more enhanced performance as compared to a randomly selected controlled which is not made for the said objective. In this paper four Java based controllers namely Beacon, OpenDaylight, Floodlight and ONOS have been taken and their suitability has been checked for the performance measuring parameters namely throughput, jitter, packet loss and latency in TCP and UDP traffic of the network. Out of these four controllers, two controllers namely ONOS and OpenDaylight are distributed controllers whereas Beacon and Floodlight are centralized controllers. All the controllers have been run in centralized manner one at a time and the performance measuring parameters have been evaluated for TCP and UDP traffic. Logic behind running distributed controllers in centralized manner is to compare their performance with pure centralized controllers if they are run in centralized manner. TCP is a connection oriented protocol and due to its property of error checking, packet retransmission, congestion control mechanism etc measurement of throughput in TCP traffic is more important than measuring it for UDP traffic so throughput has been checked in TCP traffic. UDP is connection less data transmission protocol which is basically used with the applications which send data sporadically or send stream data where small amount of data loss can be accepted. So, two parameters concerned with UPD traffic namely jitter and packet loss have been evaluated with UDP traffic of the network. It has been analysed that Floodlight is performing best for throughput in TCP traffic when the size of the network is increasing. In case of jitter and delay ODL is performing best in UDP traffic. In case of Packet Loss Beacon is best. Mininet has been used as emulator to create the tree networks of different depths. iperf has been used as TCP and UDP traffic generator.***

Keywords: SDN, Openflow, TCP & UDP, Iperf, Mininet.

Revised Manuscript Received on May 30, 2020.
***** Correspondence Author
Pinkey Chauhan*, Department of Computer Applications, NIT Raipur, Raipur, C.G., India.
Mithilesh Atulkar, Department of Computer Applications, NIT Raipur, Raipur, C.G., India.

I. INTRODUCTION

Separating out control plane from the forwarding plane is the main idea behind the development of Software Defined Networking (SDN). SDN is a new network architecture which consists of three layers namely Application layer, Control Layer, and Infrastructure layer.Some of the main advantages of separating out control plane from the forwarding plane over the traditional networking is that it provides centralized control, programmability of the network, less capital cost, less operational cost, less agility, higher flexibility, easy customization and neutrality. Due to its popularity, and scope in the field of networking it is being adopted by academia and giant IT organizations. It has been said by MIT [1] “One of the Top Ten Innovative Technologies to Change the World”.

The performance of the SDN mainly depends upon the performance of controller as controller plays the role of brain in SDN architecture. Many controllers, free and commercial are available nowadays so before using any of them it is important to evaluate its performance. There are some well known criteria those can be used to measure the performance of the SDN controller like throughput, jitter, packet loss and delay.

In this paper four Java based controllers namely Beacon [3], Floodlight [4], OpenDdaylight [5] and ONOS [6] have been taken one by one as a controller of SDN. Out of these four controllers two controllers are centralized controllers and two are distributed controllers.

Both TCP and UDP are the transport layer protocols of TCP/IP protocol suite used for communication between two systems in the network. TCP is connection oriented and UDP is connection less protocol. TCP's 3-way handshaking technique is used to establish the connection. For this TCP uses the messages like SYN, SYN-ACK, ACK whereas UDP is a connection less and hence uses no handshake message or protocols. TCP provides error checking and also does error recovery while UDP only performs error checking and does not do error recovery.

After using a controller in the control layer of SDN, TCP and UDP traffic have been generated in the network and then network performance measuring parameters namely throughput, jitter, packet loss and RTT have been evaluated. The topology in the Infrastructure layer of SDN is fixed as tree topology with varying number of depths and with fixed fanout value.

Reason of selecting tree topology is that just by going one level higher, number of nodes increases exponentially and thus by going just some higher level of depths a big number of nodes can be created and the parameters can be evaluated in a big sized network.The paper is further organized as follows: Section II describes related works, Section III shows motivation, Section IV describes controllers and their features, Section V is concerned with the implementation of the work, Result and discussion is available in Section VI. Section VII shows conclusion & future work.

II. RELATED WORKS

Four controllers namely Ryu, POX, OpenDaylight and ONOS have been comparatively studied in [7]. They have studied only 2 parameters namely RTT and bandwidth. They have observed the bandwidth of network by running the forwarding layer as hub mode and as switch mode with each controller. They found better bandwidth in Switch mode as compared to Hub mode which is obvious. ONOS is showing best performance in switch mode. ONOS is providing lowest convergence time in case of RTT. Without recommending any controller they stated it depends upon the needs and situation of the network to select the controller. They have not compared other important parameters used for comparing performance of the controllers like jitter and packet loss. Also, they have not used other Java based controllers.In [8], the performance comparison of 5 controllers namely POX, Ryu, Nox, Beacon and, Floodlight has been done using Cbench. They have only compared Latency and Throughput by using different number of switches and threads respectively. They found Beacon is performing better than other controllers. They have not used other Java based controllers and also not evaluated other network parameters. In [9], packet transmission rate, bandwidth utilization, throughput, and Round Trip Time on three topologies; linear, tree, and single have been studied using Mininet's reference controller. They are comparing the parameters in different topologies. They have not checked other important network parameter and also, they have not used any external controllers which would be the case in real scenario. In [10], POX and Floodlight controllers have been evaluated on the criteria of RTT and throughput on four topologies; linear, tree , single, and custom. They found Floodlight better than POX.In [11], Floodlight, Ryu, OpenDaylight and ONOS have been evaluated on two parameters using Cbench. In their environment, in one case they changed the number of switches and in another case, they changed the number of MACs. They stated that in case of Latency Ryu is better and in case of Throughput ONOS is better. They have not evaluated other network parameters.

In [12], two parameters; throughput and latency have been evaluated in different topologies; Linear, Single, Tree, SDN-SAT, Dumbbell, and DCN in two controllers namely POX and Ryu. In [2], The performance evaluation of three python based controllers; Ryu, Pyretic and POX has been done. RTT, Throughput and Web Server Latency were the parameters used for performance evaluation. They found Ryu is the best in all the cases.

In [13], Floodlight has been used as a controller for comparing Openflow network simulator and Emulator, Estinet and Minet respectively. They have used RTT to measure the performances of Estinet and Mininet. They have created a network like a grid of N x N size and then used ping to measure the RTT. They have neither used different controllers not topologies.

In [14], many controllers have been evaluated on the parameters throughput and latency using cbench. Apart from these two features they have also compared the controllers on different features. They have concluded that OpenDaylight is the best controller among others on the basis of different features. They claimed that OpenDaylight's capability of IoT data broker made it the first competitor in the election of the future controller.

In [15], Mininet's reference controller has been used to evaluate the throughput and packet loss in TCP and UDP traffic in three topologies; linear, single and custom tree topology. They have evaluated the parameters in different topologies. Also, they have not evaluated other network parameters like jitter and RTT.

III. MOTIVATION

In the literature survey it is found that all the related works have been done either in different topologies or changing the number of nodes in a fixed topology or by using Mininet's reference controller. No work has been done to observe the behavior of TCP and UDP traffic in Java based SDN controllers by gradually increasing the number of nodes in a network. Tree topology is the good choice for increasing the number of nodes in the network and then checking the behavior of all the mentioned controller in handling TCP and UDP traffic. This can be done just by using different value of depth and in a fixed value of fanout. By increasing the depth by one, the number of nodes can be increased exponentially, and thus real performance testing can be done on a big network having large number of nodes.

IV. CONTROLLERS AND THEIR FEATURES

In Software Defined Networking, the network is divided into 3 layers namely Infrastructure Layer, Control Layer and Application layer. It has been given in the Fig. 1.

Topmost layer of the SDN is Application layer. This is the layer from where users interact with the SDN. From this layer user give the commands to the SDN using APIs.

Middle Layer of SDN is control layer which is the most important layer of the SDN. The work of this layer is to control overall behavior of the network devices. It has the complete information of all the devices of the network, and it gives instruction to all the devices.

Bottom Layer of SDN is Infrastructure layer. It consists of all the networking devices those are concerned with forwarding the packets in the network. No device in this layer consists of any decision-making logic but it just performs the actions on the packets coming to it according to the instructions given by control layer.

The set of rules given by the users in the form of APIs are called north bound rules. These rules are used for making the communication between control layer and application layer.

The rules that are used for communication between control layer and forwarding layer are called south bound rules. Openflow is the main protocol

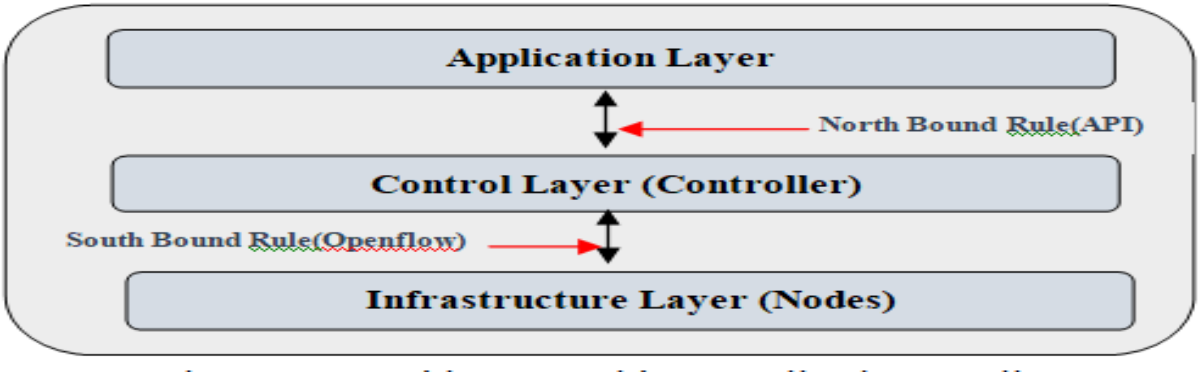

Fig.1. SDN Architecture with Centralized Controller

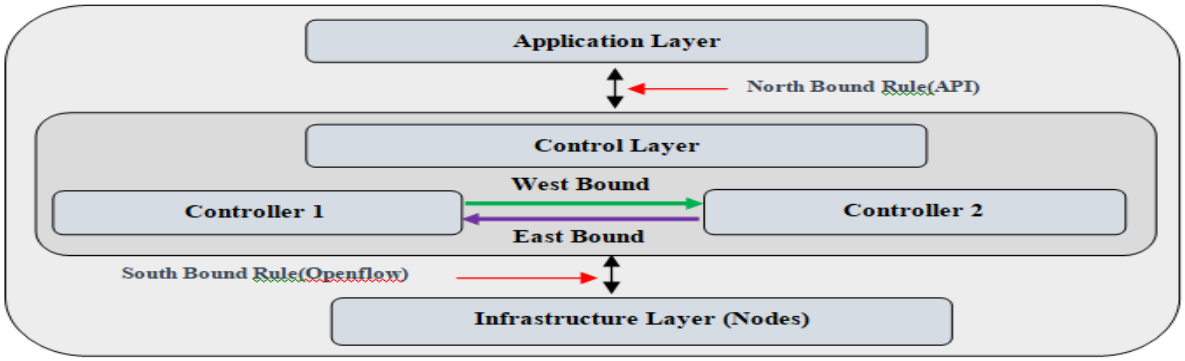

Fig. 2. SDN Architecture with Distributed Controller

Table I. Feature Comparison table of controllers

PARAMETERS	ONOS	OpenDaylight	Beacon	Floodlight
Programming Language	Java	Java	Java	Java
GUI	Web Based	Web Based	Web Based	Web/ Java Based
Documentation	Good	Very Good	Fair	Good
Modularity	High	High	Fair	Fair
Distributed/Centralized	Distributed	Distributed	Centralized	Centralized
Platform Support	Linux,MAC OS, And Windows	Linux,MAC OS, And Windows	Linux,MAC OS, And Windows	Linux,MAC OS, And Windows
Southbound APIs	OF1.0, 1.3, NETCO NF	OF1.0, 1.3, 1.4, NET-CONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	OF 1.0	OF 1.0, 1.3
Northbound APIs	REST API	REST API	REST API	REST API
Partner	ON.LAB, At&T, Ciena,Cisco, Ericsson,Fujitsu, Huawei,Intel, Nec,Nsf.Ntt Comunnication, Sk Telecom	Linux Foundation With Memberships Covering Over 40 Companies, Such As Cisco, IBM, NEC	Stanford University	Big Switch Networks
Multithreading Support	Yes	Yes	Yes	Yes
OpenStack Support	No	Yes	No	No
Application Domain	Datacenter, WAN and transport	Datacenter	Research	Campus

which is used as Southbound rule. There also exist east bound and west bound rule which are used in the case where

more than one controller is being used for running the SDN in fail safe or High Availability (HA) mode. This is also called distributed architecture of the SDN. It has been shown in the Fig. 2.Features of the controllers used in this paper are given below.

i. Beacon: Beacon is a modular, cross-platform, fast, Java-based controller that supports Openflow. It provides multithreading and event-based operations. Features of Beacon are:

- Platform Independent: Runs on multiple platform as it is developed in Java.
- Open source**:** Provided under GPL v2 and Stanford University FOSS License.
- Dynamic**:** operations of running, refreshing, stopping and installation can be handled at runtime
- Rapid Development**:** Due to Java and Eclipse it simplifies development and debugging of the application.
- Fast: It is multithreaded.
- Web UI**:** Provides easy User Interface.

ii. Floodlight: Floodlight is an Java-based, Apache-licensed, enterprise-class, OpenFlow Controller. A community of developers which includes engineers from Big Switch Networks supports it. Some of the key features of Floodlight are:

- It provides a module loading system which gives it flexibility of extension and enhancement.
- Its setup does not require much dependencies
- It is capable to provide support to wide range of physical and virtual OpenFlow switch.
- It is not only compatible with openflow enable network, but it can handle non-OpenFlow networks also.
- Provides high performance capability due to multithreading.
- It has the support for OpenStack cloud orchestration platform.

iii. OpenDaylight: OpenDaylight is a modular, scalable, highly available, extensible, and multi-protocol controller. It has the capability to support modern different type multi-vendor networks. It provides a platform for the users which enables them to write such applications that can work across a wide variety of hardware and different south-bound rules.

iv. ONOS: ONOS is an open source SDN controller for developing next generation Software Defined Network and Network Function Virtualization solutions. Key Features of are:

- High Availability & Resiliency
- Modular Software
- Performance at Scale
- Northbound Abstractions
- Southbound Abstractions
- GUI Framework & Base UI
- YANG Tool-Chain

Although all the controllers are Java based controllers but there are some differences among their features. These have been illustrated in Table-I [14] .

V. IMPLEMENTATION

Environment for Implementation: Mininet [16] [17] and controllers were run on a desktop having Operating System Ubuntu 16.04.5 LTS 64-bit. It has 8 GB RAM, AMD Processor PRO A8-8650B R7, Turbo Speed: 3.9 Ghz. Versions of the mininet, Beacon Floodlight, OpenDaylight, and ONOS are 2.2.2, 1.0.4, 1.2, 0.7.2 and 1.11.2(Loon) respectively.

VI. METHODOLOGY AND SETUP USED

Details about the topologies used and the running setup are as mentioned below:

A. Topology*:* All the controllers namely Beacon, Floodlight, OpenDaylight and ONOS were used with tree topology with different depths and fixed value of fanout. Value of fanout has been fixed to 4 for all the trees having different depths for checking their performance of all the trees. Table II shows the configurations of all tree topologies containing number of nodes and switches in all the depths.

Number of nodes(N) and that of switches (S) for depth **d** and fanout **f** in a tree can be found out using formula

$$\text{Number of Nodes}(N) = f^0 + f^1 + f^2 + \ldots\ldots f^{d-1}$$

$$\text{Number of Switch}(S) = f^d$$

Table II. Number of nodes and switches with fanout 4 having different depths

Depth(d)	Nodes(N)	Switches(S)	Total Nodes
1	4	1	5
2	16	5	21
3	64	21	85
4	256	85	341
5	1024	341	1365

A tree having depth 2 and fanout 4 would look like as given in Fig. 3. In the figure we can see that there are total 5 number of switches and 16 number of hosts and thus total 21 number of nodes. In similar manner the tree structure for different depths can be drawn.

B. Running Setup*:* Mininet was used as SDN network emulator. In Mininet above mentioned controllers were used as remote controller one by one in place of the reference controller. Mininet emulator was used to create different configurations of tree structure. Each created tree was then connected with all the controllers one at a time. When one configuration of a tree topology is connected with a controller, Iperf [18] tool with different command line arguments was used to generate the traffic. After generating

traffic and capturing them in the files, throughput, delay, packet-loss, and jitter in the network were calculated. For doing so, the most

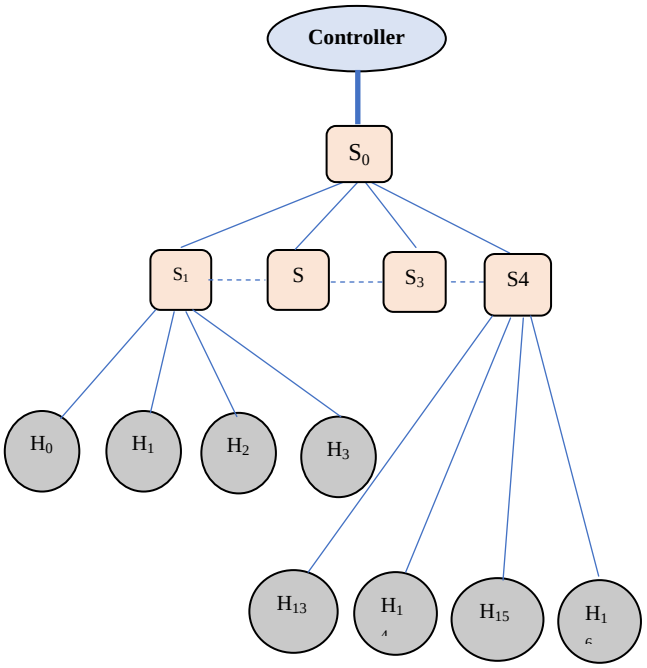

Fig. 3. Tree topology with depth 2 and fanout 4.

distant two hosts were used as iperf client and server. Like in the above shown tree H_0 was run as client and H_{16} was run as server. For calculating throughput, Iperf tool was run in TCP mode as a server in one of the two most distant nodes and it was run as client in second most distant node. The window size taken in TCP mode is 85.3 Kbytes. From the client host TCP packets were sent to server host for 20 seconds and then after performing some necessary operations on the captured TCP traffic, throughput was calculated. To calculate data loss and Jitter, iperf server was run in UDP mode by giving some necessary parameters in the command line. From the client host UDP packets were sent to the server host for 20 seconds and then after performing some necessary operations on the captured UDP traffic, jitter and packet loss were calculated. UDP buffer size was taken 208 KBytes. For calculating latency, the average time of 15 ICMP (ping) packets sent from client host to server host was taken. Xterm was used as the standard terminal emulator. Different command line arguments that were used with iperf utility to use this as client and as server are mentioned below. To generate TCP and UDP traffic from the client host Iperf was used as given below respectively.

```
#iperf -c 10.0.0.16 -p 5566 -t 20
```

```
#iperf -c 10.0.0.16 -u -b 10M  -p 5566 -t 20
```

Meaning of the arguments are shown below:
-c : used to make the host as a client machine.
-p : used to indicate the port number of the server
-t : used to mention the time for which client will run
-u : used for generating UDP traffic
-b : sets the maximum bandwidth.

To create any host as the Iperf server for TCP and UDP traffic following were used respectively.

```
#iperf -s -p 5566 -i 1
```

```
#iperf -s -u -p 5566 -i 1
```

-p : assigns the port number of the iperf server
-s : used to run iperf in server mode.
-u: Used for generating UPD traffic.
-i: is the interval on which server will measure the traffic.

In summarized manner it can be stated that for any traffic i.e. for TCP or for UDP, iperf server (IP 10.0.0.16 in the current example) was run in port number 5566 of server machine and the traffic was measured in the interval of 1 second. Similarly, from the client machine corresponding traffic was generated for 20 seconds. The measured traffic parameters were stored in the files. One more important point related to setting the UDP bandwidth to 10 Mbits per second is that, default bandwidth provided in UDP mode is 1 Mbits per second but in present age there is hardly any network having a maximum capacity of 1 Mbits per second. In almost all cases minimum bandwidth of the network is 10Mbits per second so the bandwidth is manually set to that capacity.

To measure the throughput of the generated network TCP traffic was used and to measure Jitter and Packet loss UDP traffic was used. To measure the delay (RTT) in the network ICMP/PING packets were used.

Similar kind of implementations were done for all the trees i.e. for trees of depth from 1 to 5. For a tree of any depth first host was taken as client and the last one was taken as server.

VII. RESULT AND DISCUSSION

After running the setups mentioned in implementation part following parameters from the TCP and UDP traffics were measured.

Throughput: Throughput of a computer network represents the amount of data sent from one node to another in a given time period. The unit to measure it is bits per second that might also be Kilobits per second, Megabits per second, Gigabits per second and so on. For the better performance of network throughput should be maximum. Throughput can be calculated using formula:

Throughput= Total transmitted data in bits / Total taken Time is seconds

Latency: Network latency represent the delay in the delivery of data in a network from the source to the destination. For the better performance of the network, latency should be minimal. RTT of ping packet was taken to measure the latency in the network.

Jitter: Network jitter represents the deviation in the arrival of datagrams. This becomes very important in live video or audio streaming. Jitter can be calculated using Mean packet to packet delay variation (MPPDV) approach

Jitter= mean (absolute ($d_t - d_{t-1}$)) where d_t and d_{t-1} are the delay of two successive packets reaching at destination.

Packet Loss: The amount of data that fails to reach at its destination is called packet Loss. This is directly generated by the iperf at the end of communication in UDP mode.

Following analysis was done in the TCP and UDP traffic generated in all the tree structured SDN networks with a fixed fanout value of 4 and depths from 1 to 5.

Throughput of all the controllers is decreasing while the depth of the tree is increased. It is shown in Fig. 4. It shows that while the number of nodes and switches are increasing in the network, the throughput in the network is decreasing. Second important observation is that, initially Floodlight and Beacon were comparable and giving better throughput than others but while the depth is increasing i.e. while number of nodes are increasing Floodlight is giving better throughput than Beacon also. It shows that when there is need of a reliable connection oriented big size network, Floodlight should be used.

In case of Jitter, delay and packet loss, ONOS is performing worse than other controllers when the depth exceeds value 4. It is shown in Fig. 5, Fig. 6 and Fig. 7. It implies that if ODL is used in a network where VOIP based communication is done it might create either silence in the network or overlapping of the data and thus can create problem in real communication. Here ODL might be the right choice for Jitter and Delay sensitive network while Beacon might be the right choice for the network where least packet loss is needed.

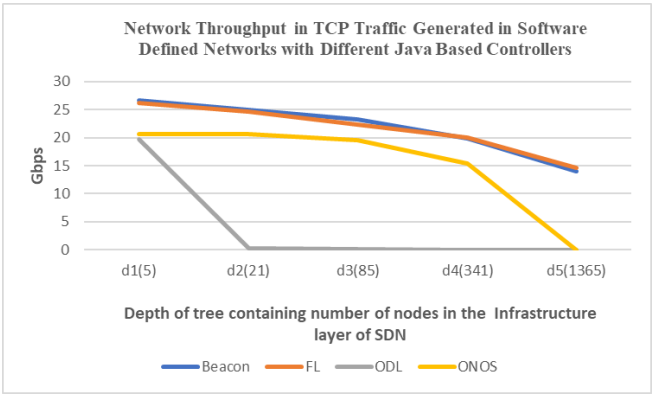

Fig. 4. Evaluation of Throughput of controllers in TCP traffic

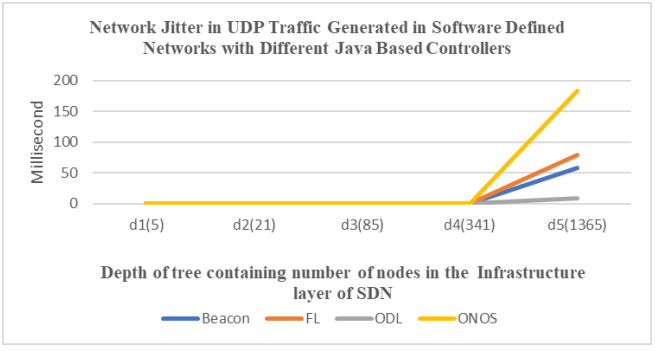

Fig. 5. Evaluation of Jitter of controllers in UDP traffic

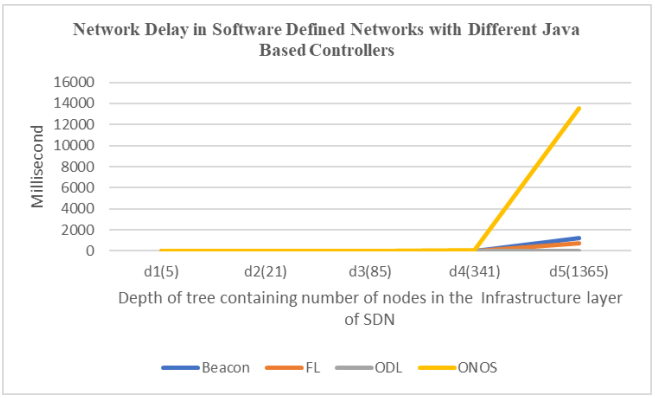

Fig. 6. Evaluation of RTT of controllers

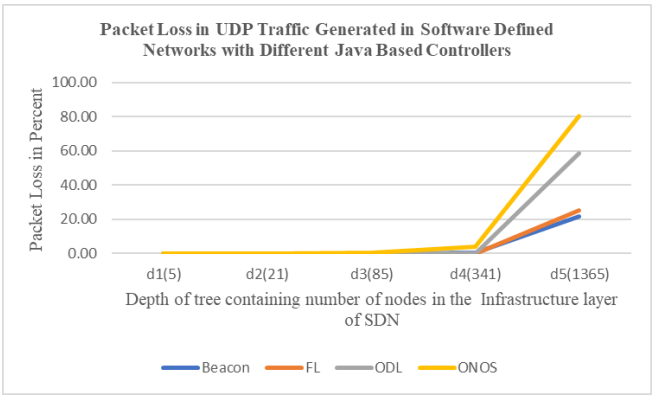

Fig. 7. Evaluation of Packet loss of controllers in UDP traffic

VIII. CONCLUSION AND FUTURE WORK

In Software Defined Networks selection of the controller cannot be done randomly but it depends upon the needs of the network for what it is being designed. If the need is to achieve maximum throughput with reliability i.e. if the network is mainly concerned with TCP traffic, then Floodlight is the best choice. If the network is mainly concerned with UDP traffic having least jitter and delay like in providing VOIP services, then ODL is the best for SDN. Beacon is the best choice where the minimum packet loss is required in big size network like in the networks which send data sporadically. Summarily Floodlight, ODL and Beacon are best for getting maximum throughput with reliability (TCP), least jitter (UDP) and delay, and packet loss (UDP) respectively. Depending upon the objective of the network being designed if a right controller is chosen, it will give more enhanced performance as compared to a controller which is chosen randomly and not performing well for the said parameter.

In current networks all the active components of networks are being used in fail safe or high availability or distributed or cluster mode by IT industries and it is also recommended for providing reliable services so in future the same setup can be tested in fail safe or high availability or distributed or cluster mode where more than one controller would have the responsibility to run the network successfully with high performance and reliability. Second work to be done in future would be to run this setup in real environment instead of an emulated environment and also this can be done in two cases,

first when only one controller is used and second when more than one controllers are working in fail safe or high availability or distributed or cluster mode.

REFERENCES

1. L. Yifan, Z. Bo ,Z. Pengyuan, F. Peiru ,L. Hui , "A Survey: Typical Security Issues of Software-Defined Networking", China Communications, Vol. 16 , issue 7 ,pp. 13 – 31, 2019.
2. Kaur, K., Kaur, S. and Gupta, V. ; Performance analysis of python based openflow controllers. 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016)., 2016, Tadepalligudem, India.
3. Beacon available at https://openflow.stanford.edu/display/Beacon/Home.html last accessed on 12/11/2019.
4. Floodlight available at http://www.projectfloodlight.org/floodlight/ last accessed on 24/11/2019.
5. OpenDaylight available at https://www.opendaylight.org/ last accessed on 10/12/2019.
6. ONOS available at https://www.opennetworking.org/onos/ last accessed on 25/12/2019.
7. Stancu, A., Halunga, S., Vulpe, A., Suciu, G., Fratu, O. and Popovici, E. (2015). A comparison between several Software Defined Networking controllers. 2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), Oct. 2015, Nis, Serbia..
8. Zhao, Y., Iannone, L. and Riguidel, M. (2015). On the performance of SDN controllers: A reality check. 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)., Jan. 2016, San Francisco, CA, USA.
9. Bholebawa, I. and Dalal, U. (2016). Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet. International Journal of Computer and Communication Engineering, 5(6), pp.419429, Nov 2016.
10. Bholebawa, I. and Dalal, U. Performance Analysis of SDN/OpenFlow Controllers: POX Versus Floodlight. Wireless Personal Communications, 98(2), pp.1679-1699, Jan. 2017.
11. Mamushiane, L., Lysko, A. and Dlamini, S. A comparative evaluation of the performance of popular SDN controllers. 2018 Wireless Days (WD). 2018 Dubai, United Arab Emirates.
12. Ali, J., Lee, S. and Roh, B. (2018). Performance Analysis of POX and Ryu with Different SDN Topologies. Proceedings of the 2018 International Conference on Information Science and System - ICISS '18. pp 244-249, Apr, 2018, Jeju, Republic of Korea.
13. S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," *2014 IEEE Symposium on Computers and Communications (ISCC)*, 2014.
14. O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, 2016.
15. M. T. Naing, T. T. Khaing, and A. H. Maw, "Evaluation of TCP and UDP Traffic over Software-Defined Networking," *2019 International Conference on Advanced Information Technologies (ICAIT)*, 2019.
16. R. L. S. D. Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014.
17. Mininet available at http://mininet.org/ , last accessed on 12/11/2020.
18. Iperf , available at - https://iperf.fr/ (Last accessed on 10-11-2019)

AUTHORS PROFILE

Pinkey Chauhan holds a master's degree in computer applications from Madhav Institute of Technology & Science, Gwalior and Master of Philosophy from School of Studies in Computer Science & IT, Pt. Ravi Shankar Shukla University Raipur. Currently she is pursuing Ph.D. from Department of Computer Applications, National Institute of Technology Raipur (Chhattisgarh), Raipur, C.G. Her area of Interest includes computer networks, network security, computer programming, software defined networking and Artificial Intelligence.

Dr. Mithilesh Atulkar, Associate Professor Department of computer Applications, National Institute of Technology Raipur (Chhattisgarh) India, has more than Twenty six year of teaching and research experience. His primary domain of teaching and research includes Artificial Intelligence, parallel processing, and Software Defined Network. He has guided more than 70 MCA projects , and currently guiding 07 Ph.D. scholar. He is a Life member of ACM ,IETE,ISTE,CSI,ORSI , IE(India) and member of some other professional bodies. He has extensive publication record in International/National journals and conferences. He has authored 04 books and many book chapters. He is reviewer of many reputed journals. He has participated in various national and international seminars, workshops and conferences. He has received "Pratibha Samman" Award by ISTE Chapter Raipur for 2001. He has been involved in various administrative responsibilities since 1994. He has served as Controller of examination for 06 years at NIT Raipur(C.G.).Presently he is holding additional charge of Registrar at NIT Raipur(C.G.).