

Experimental comparison and evaluation of various OpenFlow software defined networking controllers

Sumit Badotra*, Surya Narayan Panda

Institute of Engineering and Technology, Chitkara University, Punjab, India

ABSTRACT

In recent years, SDN has gained a lot of popularity. There is a basic principle behind the growth of SDN which states the separation of the control plane from the data plane. In the data plane, all the network devices constitute whereas in the control plane the core element is situated known as SDN controller. The controller is the integral element of SDN based network. It manages the entire network and maintains the overall functionality. In the last years, there are many SDN controllers came into existence. This paper aims to provide the experimental comparison among the seven most used SDN controllers both in research and industry. The comparison and experimentation analysis is carried out in an emulator tool known as Mininet with four different network topologies (single, linear, tree, custom) and the varied number of nodes (10, 50, 100, 500, 1000). The parameters for comparison are Round Trip Time (minimum, maximum, average) and standard deviation. This paper will prove to be beneficial for the researchers and industry people who are making use of these SDN controllers, it will help them to choose a particular controller and analyze their performance against the selected network topologies and number of hosts.

Keywords: SDN controllers, Network topology, Nodes, Performance, Analysis.

OPEN ACCESS

Received: April 28, 2020


Revised: June 3, 2020

Accepted: June 17, 2020

Corresponding Author:

Sumit Badotra

summi.badotra@gmail.com

 **Copyright:** The Author(s). This is an open access article distributed under the terms of the [Creative Commons Attribution License \(CC BY 4.0\)](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted distribution provided the original author and source are cited.

Publisher:

[Chaoyang University of Technology](https://www.chitkarauniversity.edu.in/)

ISSN: 1727-2394 (Print)

ISSN: 1727-7841 (Online)

1. INTRODUCTION

Until now, most network equipment's are configured individually by connecting to them. On one hand, this method is costly in time for big networks and in the hand subject to human error. SDN or software-defined network is an approach that allows separating the different component of the networking infrastructure (Software and Hardware) (Badotra and Singh, 2017; Papavassiliou, 2020). By separating the different component, SDN helps the network administrators to manage, optimize configure and secure network services very fast due to SDN dynamic and automated program. SDN is not dependent on any proprietary software or hardware due to its specific architecture (Badotra and Panda, 2020).

Its architecture is separated between 3 layers: application (Business Applications), control (Network Services) and infrastructure (OpenFlow Switch). Application layer communicate, via API (application programming interface) the needed resources and compartment to the SDN controller (in the control layer) (de Almeida Amazonas, 2014). The application layer can help the control layer for decision-making by collecting information from him. The control layer gets instructions from the application layer and transmits them to the infrastructure layer (Rowshanrad et al., 2014). It also gathers information's from the infrastructure layer and sends them back to the application layer (statistics, host tracks, and issues). The infrastructure layer role is to manage the forwarding data process (between devices and with the control layer). Between different layer north and south protocol is used and between devices in the same layer, west and east protocol is used. As we can see, the controllers play an essential role in the SDN

architecture (Horvath et al., 2015).

About the role of controllers and SDN controller is the application in SDN architecture which manages the flow control, it's the brain in this architecture. There are 2 types of SDN controller, SDN controller for NFV (Network function virtualization) in the data center and SDN controller to manage the switches in the network. The SDN controllers can play various roles in the SDN architecture such as allow the servers to tell the switches where the packets must be sent. There are many SDN controllers like Pox (Prete et al., 2014), ODL (Badotra and Singh, 2017), ONOS (Berde et al., 2014), Ryu (Badotra and Singh, 2019), Trema (Fernandez, 2013), Floodlight (Morales et al., 2015), and NOX (Zhang et al., 2014) which are coded in different languages (Python for Pox, Ruby for Trema, and Java for ODL, etc.) with different performances and applications.

The aim of the proposed work is to find out the differences in the performance of the different controllers with respect to the number of hosts, switches, Min time, Average time, Max time, and Mean deviation time for four different topologies (single, linear, tree, custom) on mininet testbed. For experimentation we have considered seven different popularly used SDN controllers (default or reference controller, Pox, Ryu, Floodlight, ONOS, ODL, and Trema).

The structure of the paper can be categorized into multiple sections. In section 2 related works are depicted with the research gaps. The methodology used for the experimentation is illustrated in section 3. In section 4 the results and analysis are provided. At last, the conclusion is given in section 5.

2. RELATED WORKS

In the recent time with the emergence in the field of SDN, many researchers have contributed towards the selection of the controller. In this section, related works in choosing the best controller are depicted.

Shah et al. (2013) had selected four different open-source SDN controllers for experimentation namely NOX, Beacon, Maestro, and Floodlight. The comparison was made on the basis of architectural view and memory which is shared among the multi-core machines. The comparison among Pox, Ryu, Trema, Floodlight, and ODL was made by the Khondoker et al. (2014). As per the author's selection of the best SDN controller for a network is based on Multi-Criteria Decision Making (MCDM). It is because making use of different varied properties of an SDN controller is one of the crucial tasks for the users. A procedure for the comparison and testing of the different Opensource SDN controllers is illustrated by Shalimov et al. (2013). They have made use of NOX, Pox, Beacon, Floodlight, Maestro, and Ryu for their experimentation and analysis. The parameters for the comparison were security, latency, reliability, throughput, latency, and scalability. Al-Somaidai et al. (2014) illustrated a discussion on five different versions of OpenFlow based

switch having different versions such as 1.0, 1.1, 1.2, 1.3 and 1.4. In order to perform the experimentation 4 different platforms (simulation and emulation) were used. 7 different types of controllers including NOX, Pox, Floodlight, ODL, Ryu, Mul and Beacon were considered.

Rowshanrad et al. (2016) evaluated the performance of the Floodlight and ODL. They had considered network QoS parameters. They had evaluated latency on the basis of three modes. One mode is having low network load the second one is having a mid load network and the last one is comprised of a heavy load network. For this, they had performed the experimentation on three different network topologies (single linear and tree).

Vishnu priya and Radhika (2019) had compared the performance of some of the popularly used OpenFlow controllers like NOX, Pox, Ryu, FloodLight and OpenFlow reference controller. The parameter for the comparison was based on their ability of data packet handling capacity. The authors had varied the different packet sizes, a number of packets, and patterns of arrival.

Badotra and Panda (2019) had compared the ODL and ONOS on the basis of burst rate, bandwidth, Round Trip Time (RTT) and throughput. They had made use of mininet emulation tools and Wireshark to capture the real data packets in the network.

The performance analysis of popular SDN controllers with respect to different topologies and a number of hosts/ switches used in the network is still an open problem. In the proposed work we have analyzed the efficiency of the different controllers with respect to aforementioned parameters.

3. METHODOLOGY

In order to evaluate the performance of the different controllers, we have executed the experimentation by using the Mininet emulation tool. It is a tool that creates the virtual environment for different machines. It is very helpful in running the different network topologies within a few seconds. Every time for new network topology and different SDN controller we have executed the cleanup process for the experiment. This will help to avoid any previous logs and cache data.

The methodology of the experimentation is depicted in Fig. 1. We have compared the performance of seven different SDN controllers which are illustrated as follows:

- *Reference Controller*: It is a default SDN controller in the mininet. It is also known as the reference or inbuilt Mininet controller. With the help of reference controller we can run different network topologies (default and custom).
- *Trema*: The SDN controller which includes a framework in which we can create any element by making use of programming languages such as Ruby and C.

- **Floodlight:** It is an SDN controller which makes use of Java programming language to run its functionality. The support for a higher number of network routers, switches and other elements are there. This controller can sustain with both OpenFlow and non OpenFlow protocols.
- **Pox:** It is an open-source SDN controller developed in Python. One the advantage of Pox is to be easy to set up (install and run). Pox controller is already installed with the official Mininet Virtual Machine.
- **Ryu:** It is another OpenFlow SDN controller which is based on component rich functions. It is expanded by NTT labs. It is comprised of multiple defaults functionalities such as user isolation, network topology visibility, and support for different customized network controlled applications.
- **ODL:** ODL or Open Daylight, founded on the 8 of April 2013 is an open-source SDN the controller developed in Java. The project is as part of The Linux Foundation and it's the larger open-source, SDN controller.
- **ONOS:** ONOS or Open Network Operating System, founded in 2014, is also an open-source SDN controller, written in Java and like ODL is a part of The Linux Foundation collaborative project. The advantage of this controller is high performance and availability and scalability.

About the topologies: In Mininet (Network Emulator for SDN), Network topologies are created with the command “sudo mn”, the created hosts switch (OpenFlow switch), controller (Pox, Ryu, Trema) and computer act and works like real devices. Mininet permits the creation of different topologies, from 1 switch to n switch with multiple host and various link. It allows the creation of custom topology as well. During this experimentation, we used 4 topologies: Single, Linear, Tree and Custom topology, and a different number of hosts: from 10 to 1024 host. Single: The default topology, it incorporates 1 OpenFlow switch connected and N host connected to the switch. The switch is connected to the controller. To create a single topology: “sudo mn – topo = single, N” where N is the number of the host. Linear: Linear topology incorporates N switch for N host: each host is connected to 1 switch and each switch is connected to the controller. To create a linear topology: “sudo mn – topo = linear, N” where N is the number of host/switch. Tree: Tree topology has 1 switch and others are linked to it based on a fanout number. For example, a fanout number of 5 means 5 OpenFlow switch with 5 hosts connected to every switch. To create a tree topology: “sudo mn – topo = linear, depth = x, fanout = y”. Custom: Custom topology are created in Python; custom topology allows you almost everything like 2 switches and only 1 host etc. In this experimentation we have switch linked together and all the host are connected to the 2 switches. To launch a custom topology: “sudo mn – custom topology_name.py – topo my topo”.

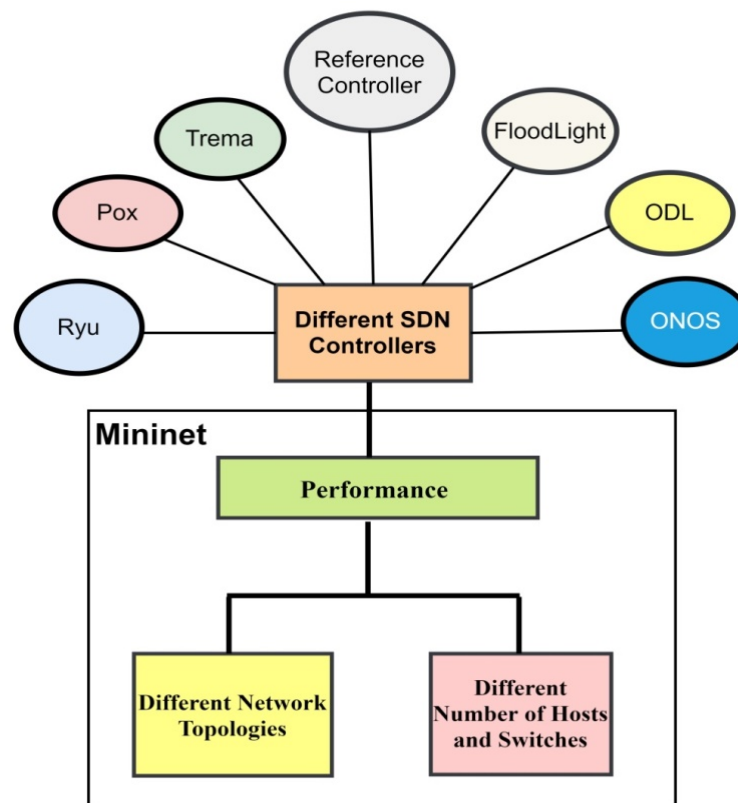


Fig. 1. Methodology

4. RESULT AND ANALYSIS

In this section analysis of 4 different attributes (minimum time, maximum time, average time and standard deviation) with respect to 4 different network topologies (single, linear, tree and custom) is illustrated. The comparison and analysis are carried out in a virtual environment of the Mininet emulation tool. The virtual machines are comprised of 32 bit Ubuntu machine 16 GB RAM and i7 processor. The mininet machine is connected to the other machines (where controllers are placed) through a layer 2 virtual switch.

Analysis of minimum RTT: In Fig. 2 it can be clearly seen that minimum RTT among all controllers against the rate of the increased number of hosts is taken by ODL. This indicates the good performance of a controller which takes less time to respond with the increased number of hosts. In

single topology for 10 numbers of hosts, ODL is taking 0.024 ms, whereas for 1000 number of hosts it is taking only 0.023 ms time to establish a connection that is less than other controllers. In the case of linear topology, ODL is taking 0.029 ms time against 1000 hosts whereas for 10 numbers of hosts it is taking 0.025 ms. In tree topology, it can be observed that for 9 a number of hosts 0.032 ms time and on the other hand for an increased number of hosts (1024), it is taking only 0.031 ms.

In case of custom topology having for 10, 1000 number of hosts, ODL is taking minimum time i.e. 0.035 ms and 0.018 ms. This clearly indicates that ODL takes the minimum time to establish the connectivity among the hosts. With the increase in the number of hosts, the minimum time is taken by the ODL controller. This makes the ODL perform better in the aforementioned topologies and other parameters as shown in Fig. 2.

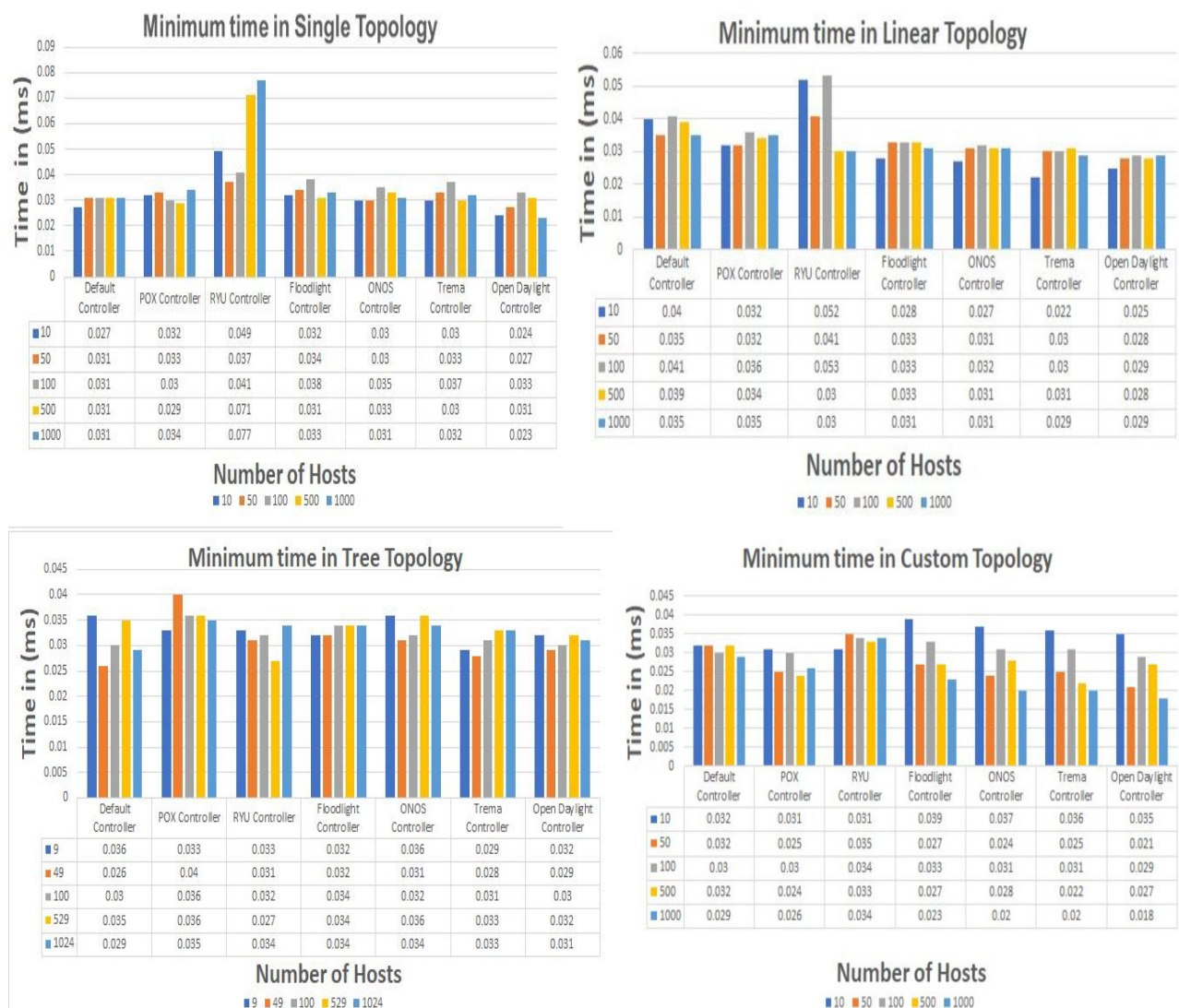


Fig. 2. Minimum time analysis

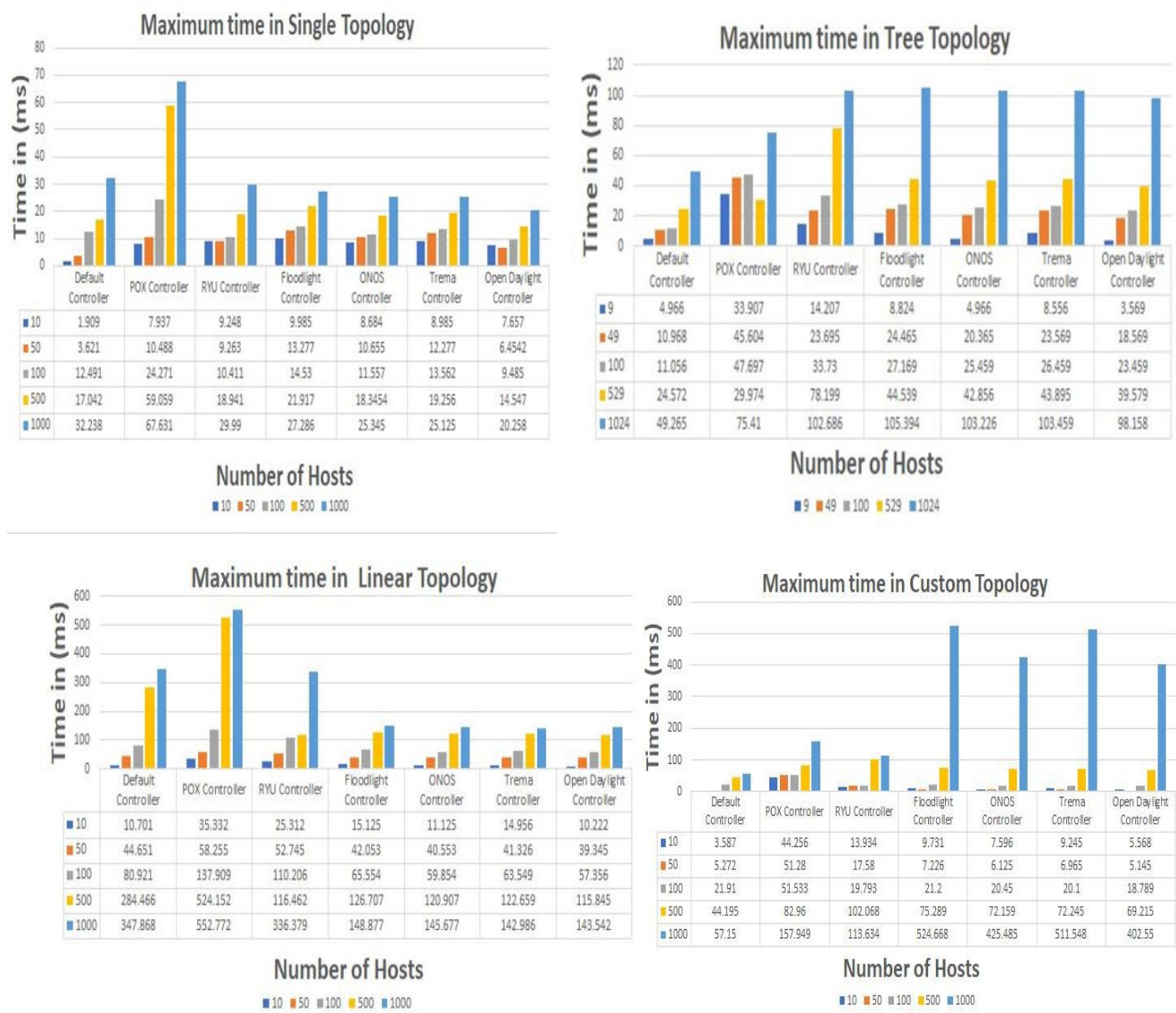


Fig. 3. Maximum time analysis

Analysis of maximum RTT: In Fig. 3(a), the maximum RTT taken by the controllers against the varied number of hosts is illustrated. The maximum RTT taken by the SDN controllers is depicting the extreme time taken by the source and destination hosts to establish a communication. It is one of the important parameters to evaluate the performance of a controller. In single topology, for 10 numbers of hosts ODL controller is taking the maximum RTT of 7.657 ms and for 1000 hosts Pox controller takes 67.631 ms RTT. For 9 hosts in a tree topology, the maximum RTT is 33.907 ms which is happened in the case of the Pox controller. On the

other hand, when the number of hosts is increased to 1024 the maximum RTT is achieved by the Floodlight controller in case of tree topology.

When the same scenario mentioned above is executed in case of linear and custom topology, it is observed from Fig. 3(b) that for 10 numbers of hosts the maximum RTT is 35.332 ms, 44.256 ms which is happened in the case of the Pox controller. When the numbers of hosts are increased to 1000 in case of both the topologies the maximum 552.772 ms (Pox) and 524.668 ms (Floodlight).



Fig. 4. Average time analysis

Analysis of average RTT: In Fig. 4 the average RTT analysis of all the selected topologies against the different SDN controllers is depicted. In every case, the average RTT

varies. It illustrates the varied functionality among the number of hosts and the type of network topology. The average time is directly proportional to the number of nodes in the network and used network topology.



Fig. 5. Standard deviation analysis

Analysis of standard deviation (mdev is a standard acronym used for standard deviation): In Fig. 5 the standard deviation of ping times for all the selected SDN controllers. Like other experimentation procedure done earlier, numbers of hosts are ranging from 10-1000 and Fig. 5 illustrates the time in ms against the seven selected parameters.

5. CONCLUSIONS

With the varied availability of multiple SDN controllers, it creates a lot of ambiguity to select the best controller. In this paper, we have evaluated the performance analysis of seven SDN controllers in a network prototype emulator (Mininet). For evaluation, one of the crucial parameters was RTT. From experimentation, we have evaluated that different controller performs differently and their performance is directly proportional to the network load (Number of hosts, switches). For real network, when aforementioned controllers are chosen this research work would prove to be beneficial for various network administrators to analyses the performance.

REFERENCES

- Al-Somaidai, M.B., Yahya, E.B. 2014. Survey of software components to emulate OpenFlow protocol as an SDN implementation. *American Journal of Software Engineering and Applications*, 3, 74–82.
- Badotra, S., Panda, S.N. 2019. Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking. *Cluster Computing*, 1–11.
- Badotra, S., Panda, S.N. 2020. Software-defined networking: A novel approach to networks. In *Handbook of Computer Networks and Cyber Security* 313–339. Springer, Cham.
- Badotra, S., Singh, J. 2017. A review paper on software defined networking. *International Journal of Advanced Research in Computer Science*, 8.
- Badotra, S., Singh, J. 2017. Open daylight as a controller for software defined networking. *International Journal of Advanced Research in Computer Science*, 8.
- Badotra, S., Singh, J. 2019. Creating firewall in transport layer and application layer using software defined networking. In *Innovations in Computer Science and Engineering*, 95–103. Springer, Singapore.

- Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Parulkar, G. 2014, August. ONOS: towards an open, distributed SDN OS. In Proceedings of the third workshop on Hot topics in software defined networking, 1–6.
- de Almeida Amazonas, J.R., Santos-Boada, G., Solé-Pareta, J. 2014, July. A critical review of OpenFlow/SDN-based networks. In 2014 16th International Conference on Transparent Optical Networks (ICTON), 1–5. IEEE.
- Fernandez, M. 2013, January. Evaluating OpenFlow controller paradigms. In ICN 2013, The Twelfth International Conference on Networks, 151–157.
- Horvath, R., Nedbal, D., Stieninger, M. 2015. A literature review on challenges and effects of software defined networking. *Procedia Computer Science*, 64, 552–561.
- Khondoker R, Zaalouk A, Marx R, Bayarou A. 2014. Feature-based comparison and selection of software defined networking (SDN) controllers, World Congress on IEEE on Computer Applications and Information Systems (WCCAIS), 1–7.
- Morales, L.V., Murillo, A.F., Rueda, S.J. 2015, September. Extending the floodlight controller. In 2015 IEEE 14th International Symposium on Network Computing and Applications, 126–133. IEEE.
- Papavassiliou, S. 2020. Software defined networking (SDN) and network function virtualization (NFV).
- Prete, L.R., Shinoda, A.A., Schweitzer, C.M., de Oliveira, R.L.S. 2014, June. Simulation in an SDN network scenario using the Pox Controller. In 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 1–6. Ieee.
- Priya, A.V., Radhika, N. 2019. Performance comparison of SDN OpenFlow controllers. *International Journal of Computer Aided Engineering and Technology*, 11, 467–479.
- Rowshanrad, S., Abdi, V., Keshtgari, M. 2016. Performance evaluation of SDN controllers: Floodlight and OpenDaylight. *IIUM Engineering Journal*, 17, 47–57.
- Rowshanrad, S., Namvarasl, S., Abdi, V., Hajizadeh, M., Keshtgary, M. 2014. A survey on SDN, the future of networking. *Journal of Advanced Computer Science & Technology*, 3, 232–248.
- Shah, S.A., Faiz, J., Farooq, M., Shafi, A., Mehdi, S.A. 2013, June. An architectural evaluation of SDN controllers. In 2013 IEEE International Conference on Communications (ICC), 3504–3508. IEEE.
- Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., Smeliansky, R. 2013. Advanced study of SDN/OpenFlow controllers. Proceedings of the 9th Central & Eastern European Software Engineering Conference, ACM, 1–6.
- Zhang, X., Hou, W.G., Han, P.C., Guo, L. 2014. Design and implementation of the routing function in the nox controller for software-defined networks. In *Applied Mechanics and Materials*, 635, 1540–1543. Trans Tech Publications Ltd.