Contents lists available at: http://qu.edu.iq

Al-Qadisiyah Journal for Engineering Sciences

Journal homepage: https://qjes.qu.edu.iq

Research Paper

An empirical mathematical model to select the best controller in software-defined networks infrastructure

Alaa Taima Albu-Salih [1], *Osama Majeed Hilal* [1], *Ahmed Al-Shammari* [1] *and Mehdi Naseriparsa* [2]

[1] *Department of Computer Science, Faculty of Computer Science and Information Technology, University of Al-Qadisiyah, Al Diwaniyah, 58002, Iraq.*
[2] *Institute of Innovation, Science and Sustainability, Federation University, University Drive, Mt Helen, Ballarat, Australia.*

ARTICLE INFO

Article history:
Received 28 April 2024
Received in revised form 04 July 2024
Accepted 15 July 2025

keyword:
SDN
Mininet
Mathematical Model
Ryu
POX
ODL
Floodlight

ABSTRACT

Software Defined Networks (SDNs) are one of the most important modern technologies in the field of networks, because of their advantages in the architecture and management of networks and control of their full functionality. SDN is distinguished from traditional networks by the presence of a central control element, which is the controller that is responsible for all operations that occur in the network. The controller is the main element that determines the success or failure of software-defined networks, so it was necessary to study and compare the different types of controllers that exist today. This paper proposes an empirical mathematical model to choose the best controller for SDN by using a Mininet emulator, concerning two performance metrics (Throughput and latency) for diverse parameters such as different types of topologies, diverse numbers of hosts, diverse numbers of switches, and diverse numbers of threads. These performance metrics have different weights depending on the needs of the users. We employ OpenFlow as a southbound protocol and five SDN controllers (Ryu, POX, OpenDaylight (ODL), and Floodlight). The results demonstrate that the suggested mathematical model is effective and flexible in choosing the best controller since the weights of performance measures are selected based on the needs of the user. The performance of the SDN network is better with ODL than with other SDN controllers.

1. Introduction

Software-defined networks (SDN) have been considered the most discussed topic in recent years [1]. It provided many advantages and contributed to solving many of the problems that traditional networks suffered from [2]. Businesses now look to SDN to improve network deployment and administration by bringing cloud advantages to the table [3]. By utilizing modern tools and technologies like Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), and other cloud computing services, along with API integration with their software-defined network [3]. Network virtualization enables enterprises to become more efficient. Additionally, SDN improves flexibility and visibility. In a conventional setting, a router or switch only knows the state of the network devices that are physically adjacent to it, whether it is located in the data center or the cloud. SDN centralizes this data, enabling enterprises to see and manage the whole network and devices. Additionally, businesses can join various physical networks to form a single network or divide up various virtual networks within a single physical network [4]. SDNs depend on the process of separating data routing operations from control operations involves making routing decisions that centralize network control, resulting in a reduction of faults that networks may encounter. The OpenFlow protocol is the predominant SDN controller used for communication and control of switches. The OpenFlow protocol enables the switch to get routing information from the controller and then directs data packets according to this acquired information [5]. The entity architecture of SDN technology, as defined by ONF, has three layers: the application layer, the control layer, and the infrastructure layer [6], as seen in Fig. 1. The application layer encompasses the assortment of services and programs that the network offers to the user, such as routing, ACL and QoS [7]. The controller communicates with this layer via the northern interface (Northbound API) and communicates with the infrastructure layer via the southern communication interface (Southbound API) [8]. The first controller developed was in 2008 and it is a controller that does not support multi-threading. It was developed in the C++ language and was the starting point and basis for the development of other controllers [9]. The controller, acting as the network's brain, is the most important component of SDN, it issues rules that are implemented by the devices in the Data Plane layer and communicates with this layer via the protocol OpenFlow, while it communicates with the application layer via the called NorthBound Interface. Many controller types are used in SDN, however the following are the most important ones: (NOX - Ryu – POX - FloodLight - OpenDayLight - Beacon - Maestro -Trema) [10]. In this paper, we presented an empirical mathematical model that uses the Mininet emulator to choose the best SDN controller based on a number of performance metrics. The metrics selected for this performance test are Throughput and latency. Ryu, Pox, OpenDaylight, and Floodlight are the controllers used for this evaluation and network emulation. The remainder of the paper is arranged as follows: Section 2 addresses earlier work on SDN controller comparison, Section 3 briefly reviews SDN controllers (Ryu, Pox, OpenDaylight, and Floodlight), and Mininet and methodology. Section 4 shows the results of the performance test. Finally, the conclusion is presented in Section 5.

*Corresponding Author.
E-mail address: alaa.taima@qu.edu.iq ; Tel: (+964) 781-133 5883 (Alaa Albu-Salih)

https://doi.org/10.30772/qjes.2024.149238.1217

Nomenclature

M_{SDN}	The final value of SDN controller performance.	P_L	The normalized value of latency.
M	Maximum value for each performance metric.	P_B	The normalized value of throughput.
PN	Normalized value.	$SDNs$	Software defined networks.
P	the original value of the performance metric.	α, β	The weights of throughput and latency.

2. Related work

In recent years, there have been several attempts at selecting the best SDN Controller. Related works in selecting the best controller are presented in this section. In [11], the performance of four SDN controllers (Ryu, Floodlight, ONOS, and OpenDayLight) has been evaluated through latency and throughput using Cbench. The results showed that OpenDayLight is more feature-rich in terms of interface vendor support, ONOS has the best throughput and Ryu has the best latency. In [12], the authors compared the following SDN controllers: NOX, Beacon, POX Floodlight, Ryu, OpenDayLight, and ONOS. with based statistical methods (BWM), Fig. 1. The results show that ONOS and OpenDayLight are the best controllers, while POX and NOX are the worst.

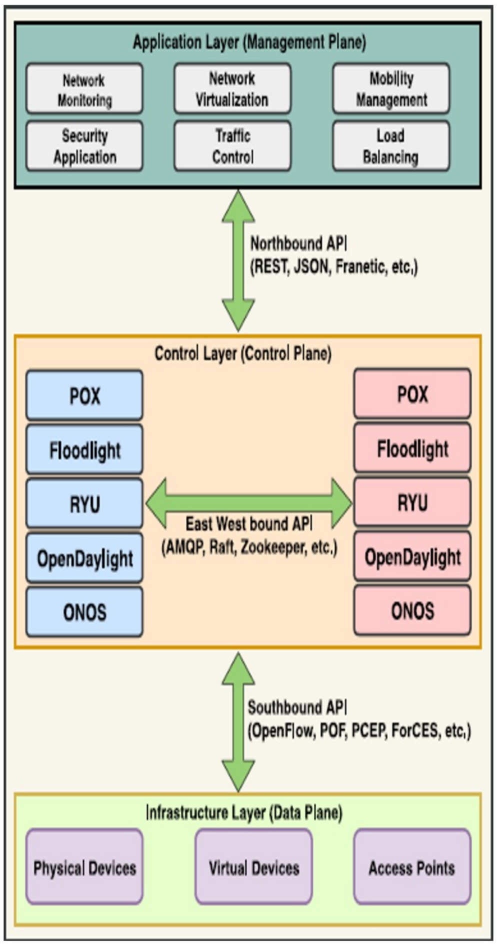

Application Layer (Management Plane)
Network Monitoring
Network Virtualization
Mobility Management
Security Application
Traffic Control
Load Balancing
Northbound API (REST, JSON, Frenetic, etc.)
Control Layer (Control Plane)
POX
Floodlight
RYU
OpenDaylight
ONOS
East West bound API (AMQP, Raft, Zookeeper, etc.)
Southbound API (OpenFlow, POF, PCEP, ForCES, etc.)
Infrastructure Layer (Data Plane)
Physical Devices
Virtual Devices
Access Points

Figure 1. The SDN architecture.

In [13], the authors evaluated the performance of FloodLight, OpenDayLight, and Ryu controllers in terms of latency, throughput, and scalability. The results show that the OpenDayLight controller has the highest performance. In [14] they compared seven widely used SDN controllers in research and industry across different network topologies and different numbers of nodes. In [15], the authors estimated the performance of five SDN controllers (libfluid, ONOS, OpenDaylight, POX, and Ryu). The results show that libfluid and POX controllers have the highest throughput. In [16], four SDN controllers' performance has evaluated (libfluid, OpenDaylight, POX, and Ryu) is evaluated based on latency, bandwidth, jitter, datagram loss, etc. The results show that Floodlight outperforms SDN new controllers. The authors in [17], proposed TOPSIS with entropy weights to select the best controller among the four selected SDN controllers (Ryu, ONOS, OpenDaylight, and POX) The outcomes exhibited that The Ryu controller is the best. In [18], a comparative tool is used to compare the performance of different SDN controllers (OpenDaylight, ONOS, RYU, Floodlight, POX, and Trema). The results of latency and throughput show that OpenDayLight, ONOS, and Floodlight outperform Ryu, POX, and Trema. In [19] the authors investigated Python-based controllers (Ryu, POX, and Pyretic). The results showed that the Ryu controller outperforms the POX and Pyretic controllers. Java-based controllers (OpenDaylight and Floodlight) are tested and compared in [20] in terms of latency and packet loss with different topologies and network loads. The results showed that Floodlight can outperform OpenDaylight in terms of packet loss and OpenDaylight can outperform Floodlight in terms of latency. While in [21] the authors examined Python-based controller (POX) and Java-based controller (Floodlight) in different network topologies. The results showed that Floodlight works faster than POX. Different Python and Java-based controllers are considered and compared in [22–26] in terms of different performance metrics. Table 1 presents a summary of the existing articles that used to evaluate the performance of several SDN controllers.

Table 1. An overview of existing studies.

Ref.	SDN Controllers	Selection Technique	Performance Metrics
Mamushiane et al. [11]	Floodlight, Ryu, ONOS, and OpenDayLight	Cbench	Throughput and Latency
Amiri et al. [12]	NOX, POX, Beacon, Floodlight, Ryu, ODL, and ONOS	MCDM	Throughput
Mendoza et al. [13]	FloodLight, OpenDayLight and Ryu	Cbench	Throughput and Latency
Badotra et al. [14]	Trema, POX, Floodlight, Ryu, ODL, and ONOS	Cbench	Minimum RTT
Abdullah et al. [15]	libfluid, ONOS, OpenDaylight, POX and Ryu	Iperf	Throughput and Latency
Mittal [16]	Floodlight, POX, and OpenDaylight	—–	Latency, Bandwidth utilization, Jitter, and Packet loss
Zobary et al. [17]	POX, Ryu, ONOS, and OpenDaylight	TOPSIS	Average RTT
Haggag et al. [18]	POX, Ryu and Pyretic	—–	RTT, Throughput and Latency
Rowshanrad et al. [17]	Floodlight AND OpenDaylight	—–	Latency, Jitter, and Loss
Bholebawa et al. [21]	POX and Floodlight	—–	Throughput and Round-trip delay

3. OpenFlow protocol (OF)

The OpenFlow (OF) protocol, sometimes referred to as the Southbound Interface, manages all communication between network devices and the SDN controller. The protocol is open-source and was created in 2009 by the Open Network Foundation (ONF) based on the v1.0 version of OpenFlow (OF). We have made improvements to the protocol, leading to the current version, OpenFlow v1.5. The upcoming release will be OpenFlow v2.0 [27]. The OpenFlow (OF) protocol serves as the key control mechanism in Software-Defined Networking (SDN) since it governs the behavior of switches in the network and enables external devices like controllers to manipulate the flow of data inside the network. Many manufacturers have started producing switches because these switches feature tables that display the entrance and exit pathways (ingress, egress) for each package on the switch. The controller can retrieve information from these tables via the OpenFlow protocol. The table input is transmitted to the OpenFlow switch, and the controller utilizes a secure channel to implement updates [28].

4. SDN controllers

The controller is the most important component in SDN, as it represents the brain of the network, and it issues the rules that are implemented by the devices in the Data Plane layer and communicates with this layer via the OpenFlow protocol, while it uses the Northbound Interface to communicate with the application layer, several SDN controllers use OpenFlow or other protocols in their operation [29]. The programming language used to program these controllers makes them unique from one another, and the version of the OF protocol that they support, in addition to other technical advantages. Due to the limitation of the size of this article, this section will only present a very small initial description of the most open-source controllers [30].

4.1 RYU

Ryu is an open-source SDN controller written in Python used to increase network resilience by making it easier to handle tasks. The Ryu controller provides a variety of components with a complete software interface that allows developers to create new ways to manage the network and build control applications with easier connectivity [31]. RYU supports most OpenFlow protocol versions from 1.0 to 1.5. It supports the STP protocol, thus preventing loops in the network. It does not support Multithreading in its work.

4.2 POX

POX is an open-source controller programmed in Python to develop control applications in SDN networks. The POX uses OpenFlow and OVSDB to provide a working environment that allows communication with SDN switches. POX components can be invoked directly through the command line where the desired functionality of the network is achieved through the use of these components. The POX controller can be used as a main controller in SDN networks because it allows fluidity to load network components [32]. Most research in the field of SDN deals with the POX controller to implement several applications such as load balancing, multimedia applications, file transfer, web servers, and other applications, which allows for continuous improvement of network performance.

4.3 OpenDaylight (ODL)

It is an open-source community project written in Java whose main goal is to develop SDNs by providing features and supporting new protocols in the controller industry. The ODL controller supports the OpenFlow protocol and can also support some other SDN standards [33]. The OpenDaylight controller is the mastermind of the SDN network, as it monitors and manages the network and generates appropriate routing rules for each network state to send them to the appropriate switches, which it stores within its flow tables. Based on the queries it requests from the switches, it adjusts the access rules or changes the flow tables. Where the network management process in SDN differs from traditional networks, is that SDN networks centralize management in the controller, and therefore the switches do not know anything about the rest of the network, they only receive commands from the controller.

4.4 Floodlight

It is an open-source controller licensed by Apache that uses the Java programming language and is developed by Big Switch Networks. A controller provided by Big Switch that supports the OpenFlow protocol in various versions. It is open-source and built in Java. It provides an advanced REST API that helps control all the functions of the controller and display its parameters on a web page. When the controller is running, many applications written in Java are run with it. The services provided by the REST API run on port 8080, and any application written in any programming language can access and display controller information and execute specific commands [34]. The controller includes many different software modules that meet the requirements of the SDN network, such as network discovery - device management - network topology manager - link discovery - routing, etc. These components communicate with each other through programming interfaces that implement them.

5. Mininet emulator and CBench

Employing an emulator, a Mininet may build a virtual network including hosts, switches, controllers, and links. Standard Linux software can be run on Mininet hosts, and the OpenFlow protocol can be supported by its switches to provide extremely flexible dedicated packet routing. Mininet supports R&D, learning, prototyping, testing, debugging, and other chores by providing a comprehensive network on a laptop or any other device. The CBench program was utilized to quantify the efficiency of the controller based on a predetermined count of switches and hosts.

5.1 Mininet

It is a well-known network emulator used to test and implement software-defined networks. This emulator has made it possible to create different topologies from virtual host statistics, links, and switches. In addition to the command line environment, Mininet also has a graphical environment called Miniedit, which can be graphically drawn in the Mininet environment to draw its topologies and how the switch, controller, and various hosts are located in this topology [6].

5.2 CBench

The most popular tool for evaluating controller performance in SDNs is Cbench. Throughput and Latency are the two main modes of criteria for this tool, and Cbench evaluates controller performance using these two criteria [35]. In latency mode, Cbench transmits a packet and then waits for a response to ascertain the processing time of the controller. When operating in throughput mode, the controller has a maximum capacity to process a specific number of packets, as indicated by transmission [36].

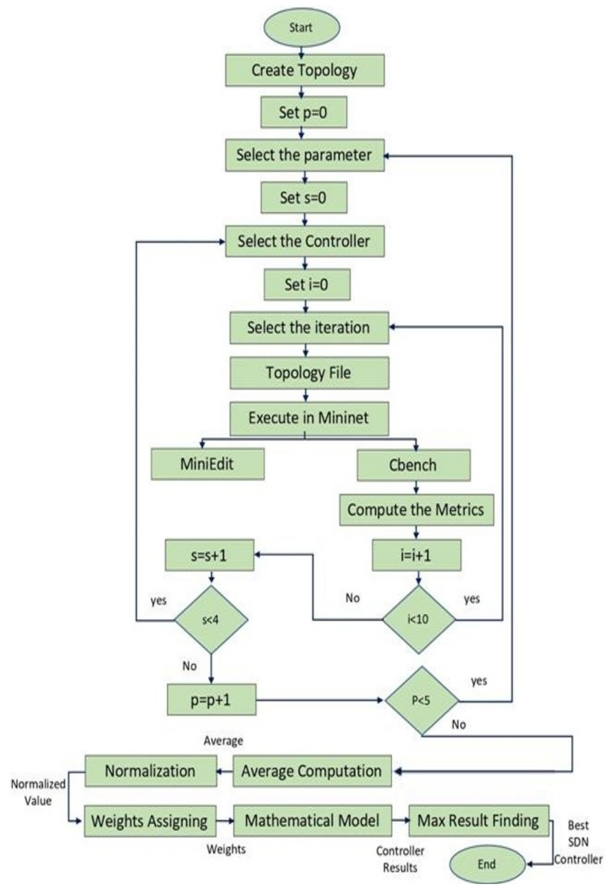


Figure 2. Implementation stages of the best SDN controller.

6. Performance metrics

In this section, performance metrics used to select the best SDN controller have been discussed.

- **Throughput**
  The throughput is usually defined as the rate at which the flow request is processed by the controller per unit time. The throughput is often: response/sec. The CBench tool is used to measure throughput.
- **Latency**
  It is the time between the moment a packet is sent from the switch to the controller and the moment the response is received from the controller to the switch., measured in seconds. It can also be measured with a CBench tool.

7. Best SDN controller selection methodology

In this section, we suggested an empirical mathematical model to select the best controller from several SDN controllers (Ryu, POX, ODL, and Floodlight) used in this paper to several performance metrics (Throughput and latency) according to many parameters. To select the best controller, the following steps can be followed:

7.1 Average computation

The first step is to compute the final average for the performance metrics for all SDN controllers.

7.2 Normalization

The second step is to apply the normalization method on the final average for the performance metrics due to some of these metrics like throughput are required to be very large while other metrics like latency want to be very small. The normalization method is done in the following steps, Eq. 1:

- Find the maximum value for each performance metric (M).
- For each metric, the normalized value (PN) was computed by dividing the original value of the performance (P) by the maximum value (M).

$$P_N = P/M \tag{1}$$

7.3 Weighting

After normalizing the final average of the performance metrics that are computed for the SDN controller, a weight will be assigned for each metric by applying the condition (the sum of this weight is 1). For example, we can assign 1 and 0 for metrics Throughput and latency used in this paper. The values of these weights are determined based on the user's needs.

7.4 Mathematical modeling

The fourth step is to employ the empirical mathematical model on the normalized values which are computed from previous steps to find the final value for the controller (MSDN) performance. Equation 2 is applied to compute the final value which indicates the value of the controller performance.

$$M_{SDN} = \alpha P_T - \beta P_L \tag{2}$$

7.5 Best controller selection

After applying the empirical mathematical to the performance of the SDN Controller, one can suggest the maximum value of all controller performance which refers to the best controller. It can be detected that the worst controller model is the minimum value of these controllers. The flow chart exemplified in Fig. 2 clarifies the stages of the selection system for the best controller. Generate the topology file created by Mininet (Using the mn command). Set $p = 0$ (p : the number of parameters). Select the parameter. These parameters include varying types of topologies, varying no. of hosts, varying no. of switches, and varying no. of threads. Set $c = 0$ (this variable to determine the no. of SDN controllers). Select the controller that is used to generate the rules that are implemented by the devices in the Data Plane layer [37]. This paper includes Ryu, POX, ODL, and Floodlight. Set $s = 0$ (no. of (iteration) loops per test). Select the no. of loops per test. Select the topology file that represents the simulation environment for SDN controllers. Execute the topology file in Mininet to perform the emulation, the output is used in Cbench and MiniEdit. Use MiniEdit as a visualization evaluation to display every event that occurs during the simulation., while the performance metrics (throughput and latency) will be calculated using the Cbench tool. Increment s by 1. If ($s < 10$) then go to step 8 (s is the number of the iteration). Otherwise, go to step 14. Increment i by 1. If $c < 4$ then go to step 6 (i is the number of the SDN controllers evaluated in this paper). Otherwise, go to step 16. Increment p by 1. If $p < 5$ then go to step 4. (p is the number of the evaluation parameters). Otherwise, go to step 18. Split the result files into several files (the number of files depends on SDN controllers that will evaluated in this paper). Compute the final average of performance metrics for all SDN controllers that will be evaluated to represent their impact on performance. Normalize the average before applying the mathematical model. Assign weight for each performance metric according to the user's need. Apply the above mathematical model (equation 2) to the resulting values from the normalization method of SDN controllers. Select the best SDN controller based on the maximum value of MSDN. After applying the empirical mathematical to the performance of SDN controllers, one can suggest the maximum value of all controllers' performance which refers to the best controller. It can be detected that the worst controller is the minimum value of these controllers.

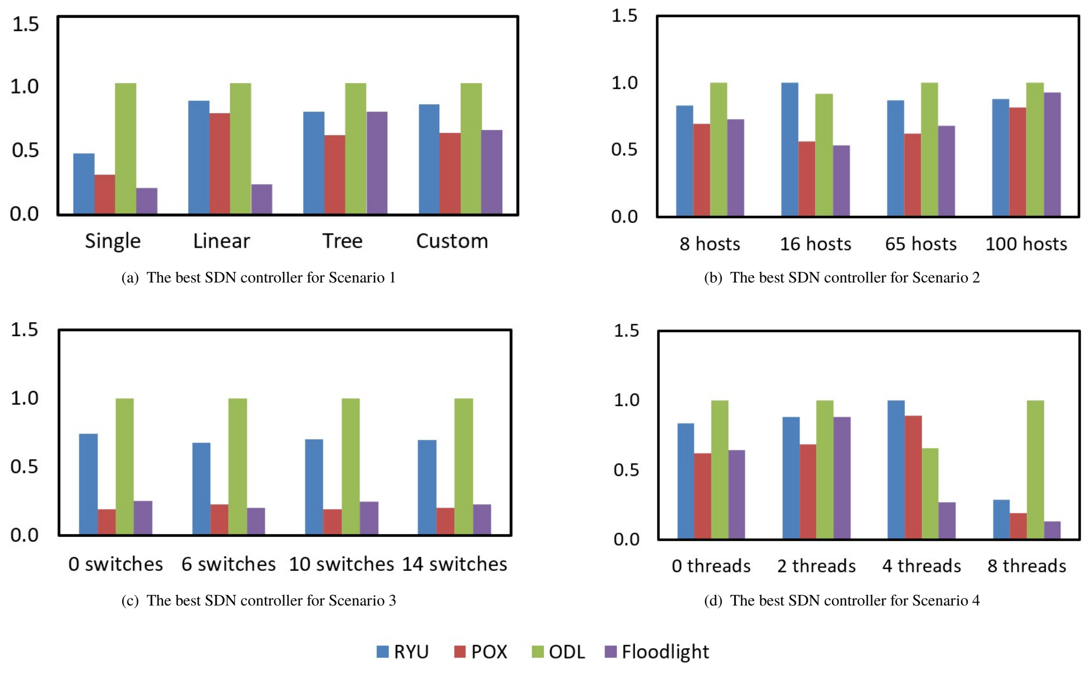
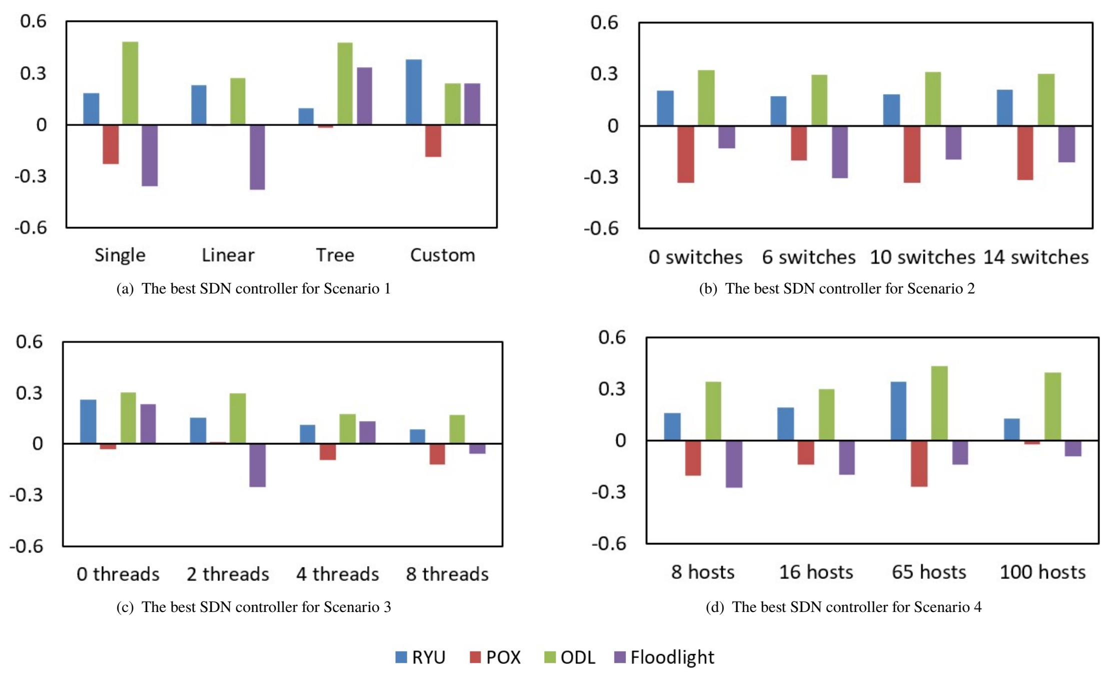


Figure 3. Histogram of the best SDN controller under case 1.

8. Experimental results

This section included the suggestion and implementation of two cases to apply this mathematical model to the scenarios listed in Table 2. The performance metrics weights utilized in this case select which SDN controllers are the best and worst. After applying the mathematical model, the values in this figure display the values of each controller. The setupinvolves using a processor Intel® Core i5 with RAM 8 GB DDR3 on macOS with a VMware Fusion



(a) The best SDN controller for Scenario 1

(b) The best SDN controller for Scenario 2

(c) The best SDN controller for Scenario 3

(d) The best SDN controller for Scenario 4

Figure 4. Histogram of the best SDN controller under case 2.

virtual machine installed on Ubuntu 21 64-bit.

Table 2. Parameters for all scenarios.

Name of Scenario	No. of Scenario	Varying Types of Topologies	Varying No. of Hosts	Varying No. of Switches	Varying No. of Threads
Varying Types of Topologies	1	Single Linear Tree Custom	8	15	1
Varying No. of Hosts	2	Linear	8, 16, 65, 100	10	1
Varying No. of Switches	3	Linear	8	0, 6, 10,14	1
VaryingNo. of Threads	4	Linear	8	12	1, 2,4, 8

8.1 Case 1

In this case, one performance metric weights 1 while the other metrics weight 0. As an illustration. The best controller for each of the four scenarios (scenario1, scenario2, scenario3, and scenario 4) in relation to case 1 appears in Fig. 3.

8.2 Case 2

In this case, the user selects the best SDN controller for each of the scenarios listed in Table 2 by giving equal weights to each performance metric. One performance metric weighs 1, while the other metrics weigh 0. The best controller for each of the four scenarios (scenario 1, scenario 2, scenario 3, and scenario 4) in relation to case 1 appears in Fig. 4.
Following the results which are obtained in cases 1 and 2 from the evaluation of SDN controllers in the Mininet emulator and Cbench to two performance metrics with various scenarios, we conclude that:
1) Although the RYU is widely used in SDN networks, however, the results of the simulation show that it is not the best among the SDN controllers.
2) The performance of the SDN network is better with ODL than other SDN controllers.
3) The SDN network has poor performance when the SDN controllers are POX or Floodlight SDN controllers.

9. Conclusion

In this paper, an empirical mathematical model is proposed to select the best controller for an SDN network using Mininet emulator and Cbench concerning two performance metrics (Throughput and latency). The suggested model is significant in different scenarios such as different types of topologies, different numbers of hosts, different numbers of switches, and different numbers of threads. These performance metrics have different weights depending on the needs of the users. We use OpenFlow as a southbound protocol and four SDN controllers (Ryu, POX, ODL, and Floodlight). After applying this mathematical model, the results indicate that this ODL was the best controller suited for SDN when compared to other controllers. In future work, we will apply Multi-Criteria Decision Making (MCDM) method such as TOPSIS to select the best SDN controller.

Authors' contribution

All authors contributed equally to the preparation of this article.

Declaration of competing interest

The authors declare no conflicts of interest.

Funding source

This study didn't receive any specific funds.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] K. Govindarajan, K. Meng, and H. Ong, "A literature review on software-defined networking (sdn) research topics, challenges, and solutions," *IEEE,In 2013 Fifth International Conference on Advanced Computing (ICoAC)*, pp. 293–299, December 2013. [Online]. Available: https://doi.org/10.1109/ICoAC.2013.6921966

[2] Y. Zhang and M. Chen, "Performance evaluation of software-defined network (sdn) controllers using dijkstra's algorithm," *Wireless Networks*, vol. 28, no. 5, pp. 3787–3800, 2022. [Online]. Available: https://doi.org/10.1007/s11276-022-03044-3

[3] O. Nafea and T. Khaleel, "An improved throttling algorithm for fog computing networks with an additional management layer," *Al-Qadisiyah Journal for Engineering Sciences*, vol. 17, no. 4, pp. 390–399, 2024. [Online]. Available: https://doi.org/10.30772/qjes.2024.146104.1089

[4] D. Ryait and M. Sharma, "Performance evaluation of sdn controllers," *Singapore: Springer NatureSingapore. In International Conference on Information, Communication and Computing Technology*, vol. 757, pp. 1009–1021, 2023. [Online]. Available: https://doi.org/10.1007/978-981-99-5166-6_68

[5] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014. [Online]. Available: https://doi.org/10.1109/JPROC.2014.2371999

[6] MiniNet. [Online]. Available: http://mininet.org/

[7] M. Al-Somaidai and E. B. Yahya, "Survey of software components to emulate openflow protocol as an sdn implementation," *American Journal of Software Engineering and Applications*, vol. 3, no. 6, pp. 74–82, 2014. [Online]. Available: https://doi.org/10.11648/j.ajsea.20140306.12

[8] W. Xia, Y. Wen, C. Foh, Niyato, and H. Xie, "A survey on software-defined networking." *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, 2014. [Online]. Available: https://doi.org/10.1109/COMST.2014.2330903

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 3, pp. 105–110, 2008. [Online]. Available: https://doi.org/10.1145/1384609.1384625

[10] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to sdn controller design," *In Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, no. 2, pp. 1–7, 2013. [Online]. Available: http://doi.org/10.1145/2535771.2535789

[11] L. Mamushiane, A. Lysko, and S. Dlamini, "A comparative evaluation of the performance of popular sdn controllers," *IEEE , In 2018 Wireless Days (WD),Dubai*, pp. 54–59, 2018. [Online]. Available: https://doi.org/10.1109/WD.2018.8361694

[12] E. Amiri, E. Alizadeh, and M. H. Rezvani, "Controller selection in software-defined networks using best-worst multi-criteria decision-making," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1506– 1517, 2020. [Online]. Available: http://doi.org/10.11591/eei.v9i4.2393

[13] D. H. Mendoza, L. T. Oquendo, and L. A. Marrone, "A comparative evaluation of the performance of open-source sdn controllers," *Latin-American Journal of Computing*, vol. 7, no. 2, pp. 64–77, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.5745007

[14] S. Badotra and S. N. Panda, "Experimental comparison and evaluation of various openflow software-defined networking controllers," *International Journal of Applied Science and Engineering*, vol. 17, no. 4, p. 317–324, 2020. [Online]. Available: https://doi.org/10.6703/IJASE.202012_17(4).317

[15] M. Abdullah, N. Al-Awad, and F. W. Hussein, "Performance comparison and evaluation of different software-defined networks controllers," *International Journal of Computing and Network Technology*, vol. 6, no. 2, 2018. [Online]. Available: http://doi.org/10.12785/ijcnt/060201

[16] S. Mittal, "Performance evaluation of openflow sdn controllers," *Intelligent Systems Design and Applications*, pp. 913–923, December 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-76348-4_87

[17] F. Zobary, "Applying topsis method for software-defined networking (sdn) controllers comparison and selection," *Communications and Networking*, vol. 210, pp. 132–141, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-66628-0_13

[18] A. Haggag, "Benchmarking and performance analysis of software defined networking controllers in normal and failsafe operations using multiple redundant controllers," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 13, pp. 5192– 5202, 2021. [Online]. Available: https://doi.org/10.17762/turcomat.v12i13.9706

[19] K. Kaur, S. Kaur, and V. Gupta, "Performance analysis of python-based openflow controllers," *3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences*, pp. 1–4, 2016. [Online]. Available: https://doi.org/10.1049/cp.2016.1515

[20] S. Rowshanrad, V. Abdi, and M. Keshtgari, "Performance evaluation of sdn controllers: Floodlight and opendaylight," *IIUM Engineering Journal*, vol. 17, no. 2, pp. 47–57, 2016. [Online]. Available: https://doi.org/10.31436/iiumej.v17i2.615

[21] I. Bholebawa and U. Dalal, "Performance analysis of sdn/openflow controllers: Pox versus floodlight," *Wireless Personal Communications*, vol. 98, no. A, pp. 1679–1699, August 2017. [Online]. Available: https://doi.org/10.1007/s11277-017-4939-z

[22] A. Bondkovskii, J. Keeney, S. van der Meer, and S. Weber, "Qualitative comparison of open-source sdn controllers," *IEEE: In NOMS 2016- 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 889–894, 2016. [Online]. Available: https://doi.org/10.1109/NOMS.2016.7502921

[23] J. Ali, B. Roh, and S. Lee, "Qos improvement with an optimum controller selection for software-defined networks," *Plos one*, vol. 14, no. 5, p. e0217631, May 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0217631

[24] S. Shamim, S. Shisir, A. Hasan, M. Hasan, and A. Hossain, "Performance analysis of different open flow-based controllers over software-defined networking," *Global Journal of Computer Science and Technology*, vol. 18(1-C), no. 1, 2018. [Online]. Available: http://creativecommons.org/licenses/by-nc/3.0/

[25] A. Aliyu, P. Bull, and A. Abdallah, "Performance implication and analysis of the openflow sdn protocol," *IEEE.In 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 391–396, 2017. [Online]. Available: https://doi.org/10.1109/WAINA.2017.101

[26] S. Islam, M. A. Khan, S. Shorno, S. Sarker, and M. A. Siddik, "Performance evaluation of sdn controllers in wireless network," *IEEE.In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, p. 1–5, 2019,May. [Online]. Available: https://doi.org/10.1109/ICASERT.2019.8934553

[27] L. Zhu, M. Karim, K. Sharif, X. Du, and M. Guizani, "Sdn controllers: Benchmarking performance evaluation," *arXiv preprint arXiv:1902.04491*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1902.04491

[28] U. Singh, V. Vankhede, S. Maheshwari, D. Kumar, and N. Solanki, "Review of software defined networking: Applications, challenges and advantages," *Springer International Publishing In Inventive Computation Technologies*, vol. 98, pp. 815–826, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-33846-6_89

[29] B. Nunes, 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. [Online]. Available: https://doi.org/10.1109/SURV.2014.012214.00180

[30] B. A. Nunes, M. Mendonca, X.-N. 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. [Online]. Available: https://doi.org/10.1109/SURV.2014.012214.00180

[31] Ryu controller, Ryu SDN Framework Community. [Online]. Available: https://ryu-sdn.org/

[32] POX controller. [Online]. Available: https://github.com/noxrepo/pox

[33] OpenDayLight controller,OpenDaylight Project a Series of LF Projects,. [Online]. Available: https://www.opendaylight.org/

[34] Floodlight controller. [Online]. Available: https://github.com/floodlight/floodlight

[35] R. Sherwood and K. Yap, Cbench: controller benchmarker. [Online]. Available: https://github.com/mininet/oflops/tree/master/cbench

[36] R. Bhavadharini and S. Karthik, "Blockchain enabled metaheuristic cluster based routing model for wireless networks," *Computer Systems Science and Engineering*, vol. 44, no. 2, pp. 1233–1250, 2023. [Online]. Available: https://doi.org/10.32604/csse.2023.025461

[37] O. Nafea and T. Khaleel, "An improved throttling algorithm for fog computing networks with an additional management layer," *Al-Qadisiyah Journal for Engineering Sciences*, vol. 17, no. 4, pp. 390–399, 2024. [Online]. Available: https://doi.org/10.30772/qjes.2024.146104.1089

How to cite this article:
Alaa Taima Albu-Salih, Osama Majeed Hilal, Ahmed Al-Shammari and Mehdi Naseriparsa. (2025). 'An empirical mathematical model to select the best controller in software-defined networks infrastructure', Al-Qadisiyah Journal for Engineering Sciences, 18(3), pp. 233-238. `https://doi.org/10.30772/qjes.2024.149238.1217`