INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

A Useful and Effective Method for Selecting a Smart Controller for SDN Network Design and Implement

Mohammed Mousa Rashid
Information Institute for Postgraduate Studies
University of Baghdad
Baghdad, Iraq
Email: phd202020555@iips.icci.edu.iq

Nadia Adnan Shiltagh Al-Jamali
Information Institute for Postgraduate Studies
University of Baghdad
Baghdad, Iraq
Email: nadia.aljamali@coeng.uobaghdad.edu.iq

Submitted: 30/11/2022. Revised edition: 31/3/2023. Accepted: 31/3/2023. Published online: 13/9/2023
DOI: https://doi.org/10.11113/ijic.v13n1-2.417

***Abstract*—Software Defined Networking (SDN) is a modern network architectural model that manages network traffic using software. SDN is a networking scenario that modifies the conventional network design by combining all control features into a single place and making all choices centrally. Controllers are the "brains" of SDN architecture since they are responsible for making control decisions and routing packets at the same time. The capacity for centralized decision-making on routing improves the performance of the network. SDN's growing functionality and uses have led to the development of many controller systems. Every SDN controller idea or design must prioritize the control plane since it is the most crucial part of the SDN architecture. Studies have been done to examine, analyze, and evaluate the relative advantages of the many controllers that have been created in recent years. In this paper, finding the perfect controller based on derived needs (for example, the controller must have a "Java" or "Python" interface), a matching process compares controller features with requirements.**

***Keywords*—Smart controller, SDN, network design, network traffic**

I. INTRODUCTION

Because it is more controllable, dynamic, and cost-effective than traditional architecture, SDN is a strong network architecture that is best for high-bandwidth applications that change quickly [1]. The idea that a network's control operations should be kept distinct from its forwarding functions became the foundation of the SDN design. This would make it simpler to directly program the network control and abstract forwarding devices for services and network applications [2, 3].

- Easy to program: Since the control function of the forwarding device has been taken away, the network's control operation can be directly programmed.
- Agile: The network administrator can change network traffic on the fly to meet different management needs since control is separated from the infrastructure underneath.
- Centralized management: Because the network's brain (controller) is logically centralized and appears as a single switch to the application and policy engines, it has a global view of the entire network.
- Employing dynamic SDN programs, network administrators may rapidly manage, modify, protect, and enhance network resources with SDN.

SDN is an emerging network paradigm that enables current network architecture constraints to be overcome; it is characterized as a foursome-pillar network architecture [4]. In a distinct data and control plane, routing decisions are flow-based rather than destination-based, the control logic is handled by an external entity known as the SDN Controller, and the network may be programmable through software applications running on top of the SDN Controller. According to the examination of the relevant literature, an attempt was made to compare the existing SDN controllers. The SDN concept and its technical components were thoroughly analysed in [5]. However, this assessment was not intended to be a comparative analysis of controllers from a commercial standpoint. Referring to aspects such as the difficulty of getting started, the Application Program Interface (APIs) that are supported, the accessibility of documentation, and the version of Openflow, among other factors, conducted a comparative analysis of SDN controllers based on a systematic study. Unfortunately, this work does not

do a comparative categorization of the offered controllers, nor does it emphasize the market viewpoint, which is crucial to the acceptability of any technology [6, 7]. Give a comparison of SDN controllers based on characteristics however, they do not perform a market-oriented comparison, such as programming language, documentation, modularity, and performance. This paper's primary contribution is a comparative analysis of the existing SDN controllers and their primary characteristics, taking into account not only functional and technical aspects, nevertheless, but market adoption, documentation availability, and OpenFlow support are also all-important factors. As a result, we provide a quantitative and qualitative comparison of the current top eight SDN controllers from both academic and industry viewpoints.

II. SDN ARCHITECTURE

Based on the prior SDN description, SDN components may be characterized as a collection of the separate data layers, control layers, and application layers that reflect the SDN architecture, as shown in Fig. 1, each of which has its own functionality and can interact through open standard interfaces. These layers were then depicted using a bottom-up approach [8].

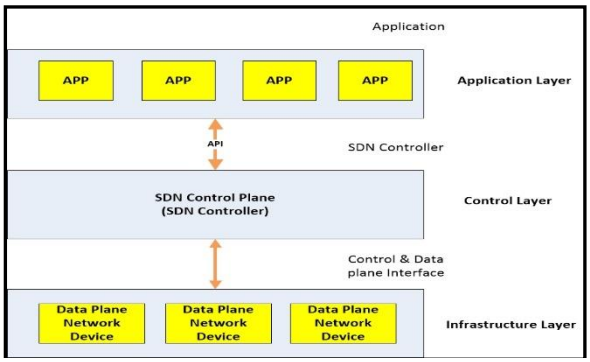

Fig. 1. Fundamental SDN components

A. Data Plane

This plane provides a description of the forwarding devices, which include switches and routers, in addition to a set of instructions that may be given via an application program interface (API). SDN network devices function similarly to traditional network devices, except those packets are forwarded based on a higher plane decision. This signifies that control is no longer delegated to an external party and is now logically centralized. The data plane and the control plane are connected through a standard interface (OpenFlow). In other words, open and standard interfaces are used to build the network brain (control) and applications (conceptually). The controller may use this interface to dynamically setup various forwarding devices. For traffic processing logic in SDN data plane forwarding devices, an API for interacting with the controller, an abstraction layer, and a traffic (packet) processing function will be implemented as software in virtual switches and as hardware in physical switches [9]. The abstraction layer is made up of one or more flow tables, and its main function is to enable the device to decide what to do with the next packet based on its contents. The packet may be routed to a particular switch port, flooded to all ports, or dropped entirely [10]. A flow table in an open flow switch is a data structure placed in a high-speed data plane data structure. It provides information about the forwarding and packet handling behaviour of the open flow switch. There are one or more flow entries in an open flow table, each with a number of components. A flow table with three entries (match fields, action, and priority), as well as a counter and timer [11, 12].

B. Control Plane

SDN controller, also known as Network Operating System (NOS), is the name given to the control plane in SDN architecture. Due to the fact that the controller is connected to all devices that perform forwarding in the bottom plane, management of the network exchange moves from distributed to centralized [2]. The controller's primary functions are as follows:

- Provide the applications plane with an abstraction view of the underlying infrastructure so they can link with devices that use the SDN (switches, routers).
- Execute the directives of the administration (load balancing, forwarding, and routing).
- Command and control all the devices that make up the network's data plane [13].

Because malfunctioning nodes are linked to the controller, the controller's logical location assists in the resolution of many distributed issues, such as quicker reactions to node or link failures. Because the controller has a full picture of the whole network, loop avoidance is substantially easier. Depending on the programming language used for implementation, there are several kinds of SDN controllers, such as the pox controller, which is implemented in the Python language, the flooding light controller, which is developed in the Java programming language, and even the NOX controller, which could use the C programming language. They're all open-source controllers, however, there are also commercial ones like HP and NEC [14].

C. SDN Application Layer

A programmable platform provided by SDN technology that enables users to build SDN applications for routing management and resolving critical network issues. Network applications communicate with the controller using an API known as the northbound interface in SDN architecture. These applications' primary function is to manage traffic within network devices by modifying flow entries via the southbound interface [15].

III. RELATED WORK

Many research aimed to compare SDN controllers. One of the first to do a comparison analysis of SDN controllers, concentrating just on controller performance in [16]. (NOX-MT, Beacon, and Maestro). "Other controllers, such as POX, FloodLight, Ryu, and OpenDaylight, have subsequently taken their place." virtualization, TLS support, open-source, GUI,

interfaces, RESTful API, documentation, productivity, platform support, modularity, OpenFlow support, age, and OpenStack Neutron support are among the criteria used in the research [17]. The modified AHP was used to evaluate five controllers (Ryu, POX, Floodlight, OpenDaylight and Trema), and "Ryu" was picked depending on their requirements, as the appropriate controller. Changing the stated scale, on the other hand, would result in a different conclusion. Advanced research on OpenFlow Controllers in SDN was conducted in this work [18]. The efficacy of NOX, Beacon, POX, Mul, Floodlight, Ryu, and Maestro which are commonly used SDN controllers, is compared. The authors utilized a program named HCP ROBE. They discovered that the evaluated controllers have some security issues, according to the results of throughput tests conducted under normal workload settings, Beacon is the controller with the best performance. As new controllers are released, this comparison must be updated to incorporate these controllers as well as additional controller functionality. The researcher examines two functioning styles in [19, 20], proactive and reactive. The proactive model performs better than the reactive mode because the rules are loaded to the switch at the start of the proactive mode, rather than each time the switch receives a packet with no matching rule in its flow table in the reactive mode. While this comparison illustrates an essential component of performance, it is insufficient to reach a decision about which controller is the best feature. When creating a new controller, [21] conducts another research that takes into account more factors. When developing a new controller, there are two distinct kinds of architectures that need be taken into consideration: shared queue with adaptive batch, and static partitioning with static batching. When compared to Maestro, NOX-MT, and Floodlight, Beacon, which uses static batching, did the best in the tests. Due to its adaptive batching design, Maestro, on the other hand, has the best latency records. As a consequence, the architecture selected is based on the behavior of the required controller with respect to its application domain. The programming languages used to develop the controllers are crucial since they are software-based. According to [22], the programming language that is used has an effect on the portability of the controller as well as its performance. [Citation needed] The authors believe that Java is the best choice since it can handle many threads and can be used on several operating systems. Python suffers from speed and stability concerns when it comes to multithreading, in contrast to C and C++, which have memory management constraints. The runtime platform is used by the network programming languages (compatibility with Linux is not supported).As a consequence, they demonstrate that among a variety of controllers (Maestro, POX, NOX, Ryu, Floodlight), the java-based Beacon has the best performance. The fact that these languages are still used today shows that they each have unique characteristics that haven't been replaced. As a result, the issue of software aging is raised in [23]. To assure the study's impartiality, the main issue investigated was a memory leak, and the evaluation was carried out using two different controllers that were based on Java (Beacon and Floodlight). The findings showed that Beacon outperformed Floodlight in terms of memory utilization. In this paper, we will examine the most popular open-source controllers based on a range of factors. This comparison is carried out with the assistance of a variety of controllers, both old and modern. The results of this comparison will make it much simpler to choose the most suitable controller for a certain application domain.

IV. SDN CONTROLLER FUNCTIONAL COMPONENTS

It's difficult to categorize SDN controllers because there are various criteria that can be used, some of which are mutually exclusive. As a result, the findings of this study are intended to enhance market and academic acceptance of SDN controllers. Since each controller design has a unique use case**,** its use is contingent not only on its ability but also on the cultural adaptation of the organization [24-26]**.**

V. ABILITIES OF SDN CONTROLLER

Management and re-programmability of networks, and data/control plane separation are SDN's key goals. The controller's components and capabilities drive a centralized model that achieves these goals. The capabilities of an SDN controller will be described in depth in the following paragraphs. The controller's role and capabilities increased as SDN progressed. To provide enterprises with more compact and effective solutions, basic qualities have been upgraded and new ones have been introduced. An SDN controller's capabilities include the following:

A. Efficiency

Efficiency refers to performance, security, and scalability. It's ideal if a controller can cover these three characteristics in the most efficient manner possible. Performance and scalability are terms used in the literature to define a controller's reaction time and the number of flows it can manage. This is a crucial trait regardless of the use case. Security may refer to a variety of functions that a controller should do in order to meet the ever-increasing number of standards. More controller implementations and versions are becoming available, and comparative studies on controller efficiency are becoming more important.

B. Support from the south

The method a controller manipulates network devices to provide optimum traffic flow has previously been described as southbound support. As was said before, there are a variety of southbound protocols that may be implemented, with OpenFlow being the most common of them. Every OpenFlow controller need to have the capacity to handle field matching, network discovery via the use of the Link Layer Discovery Protocol (LLDP), and other key features. Not only are the features of the protocol something that must be taken into consideration by implementers of southbound support, but also possible extensions, future versions, and other variables. For instance, in the case of OpenFlow, the option to communicate via IPv6 was left out of OpenFlow v1.0 but was included to the OpenFlow v1.3 standard [27].

C. Support for the Northbound

Northbound APIs offer network abstraction and programmability for customer-facing systems and applications. It's important to install a controller to handle Layer 2 and 3 and 4-7 communications. The controller should support OpenStack orchestration [28]. The controller must support vendor-specific protocols. SDN applications include firewalls, load balancers, and orchestration platforms like OpenStack. These applications may include traffic engineering or data collecting tools for network administration [29].

D. Monitoring

Another controller feature is network monitoring. The controller can discover network faults and simplify troubleshooting by using protocols (such as OpenFlow) and associated tools. The controller's benefits include thorough flow monitoring (rather than random sampling), monitoring of certain traffic classes, and so on. Standard monitoring protocols and procedures should be supported by the controller so that the data may be integrated with other management [30, 31].

E. Virtualization of Networks

The virtualizing network is the process of creating logical and virtual networks that are independent of the hardware that supports them. Virtualization includes OSI Layers 2-3 (routing) and 4-7 (load-balancing).Virtual LANs (VLANs) and Virtual Routing and Forwarding (VRF) are two instances of network virtualization that have been in use for decades (VRF) [26]. The preceding approaches are deemed restricted in scope and utility due to fast changes in network capacity, performance requirements, and other factors. SDN controllers simplify the end-to-end deployment of network virtualization, allowing enterprises to dynamically establish virtual networks and satisfy stringent criteria [32].

F. Flexibility

Another need a controller must meet is flexibility. On the one hand, it is necessary to support a wide range of applications. Controller apps, on the other hand, should employ a programming paradigm and uniform framework to guarantee that open APIs are consistent and easy to consume. This is critical for a variety of reasons, including troubleshooting and system integration [33, 34].

VI. Methodology

We followed the methods below to determine the attributes of the controllers. In the initial step, we looked for articles from journals, conferences, and workshops that have already discussed such controllers. If we discover a certain attribute and its value, we enter that information into the table. Only the websites of each controller were searched in the second step. If we located the properties and their values on the website, we inserted them into the table. This stage aided in locating more resources, such as published papers, other websites, conferences, and blogs. Technical speeches from conferences and workshops were listened to in the third step. The attributes and their values are included in the table based on the speeches and their accompanying presentations and papers published. The fourth phase involved looking at technical blogs about controllers. During the taking of notes, the rate of reaction in the blogs was also valued. In order to avoid developers from giving biased information, the attributes and their values have been confirmed and evaluated under certain conditions by comparing information from other sources.

VII. Controller Investigation

This article analyzes the eight most widely used open-source controllers: ONOS, POX, NOX, Ryu, Beacon, Maestro, FloodLight, and Trema [35]. Other SDN controllers like FlowER, MUL (C), NOX, Jaxon , NDDI-OESS , NodeFlow, and ovs-controller have not been evaluated because they are either deprecated, poorly documented certain controllers are excluded because they are designed to perform specific functions [16]. RouteFlow, Flowvisor, SNAC, Reasonance, and Oflops are other examples.

- POX is a python-based SDN controller derived from the NOX controller that is used to research SDN debugging, network virtualization, controller design, and programming paradigms [36].
- ONOS aims to "produce the Open Source Network Operating System that will enable service providers to build authentic Software-Defined Networks." Its first version, Avocet, became open-source in 2014.
- NOX (2009), originally created by Nicira Neworks, was the first SDN Controller. Concurrently, the first version of the OpenFlow protocol was published. Consequently, the first controllers to be presented were created using the OpenFlow protocol [37].
- Floodlight is a Java-based OpenFlow controller with multiple threads that was originally based on the Beacon implementation. In March 2016, its last version came out. It is meant to be a place where many different network applications can run [17].
- Ryu (Japanese for "Flow") is a component-based SDN controller backed by NTT. Ryu has a predefined set of components. These components are modifiable, extensible, and combinable in order to create a custom controller application [38].
- Beacon is a Java-based open-source OpenFlow controller that was created in 2010. It looked at new approaches to build OpenFlow controllers, with an emphasis on making them simple to use, quick, and capable of starting and stopping both new and existing applications at runtime [39-41].
- Maestro is an operating system for coordinating network control applications that was introduced in 2010 as an OpenFlow controller. Java was used to write it [41-44]. A new component may be developed using any programming language.
- Trema is supported by NEC laboratories, and its main design goals are code readability and performance.

Ruby is a programming language that is used to improve efficiency. "C" is used as a compiler language to improve performance [39].

VIII. Results of the Comparison

Comparisons were made between the controllers in terms of their available interfaces, support for virtual switching, Graphical User Interface (GUI), support for a programming language, modularity, operating system support, maturity, TLS support, and OpenStack networking support, as well as their productivity in terms of the speed at which they can write codes, the performance of their code, and the performance of their code. Table I depicts the outcome of the comparison. It may be inferred that none of the controllers are ideal when all of their qualities are considered.

TABLE I. Attributes comparison of the most widely used open-source SDN controllers

Controller **Features**	**ONOS**	**NOX**	**POX**	**Ryu**	**Beacon**	**Mestro**	**Flood-Light**	**Trema**
Programming Language	Java	C++	Python	Python	Java	Java	Java	C Ruby
Year	2014	2009	2013	2013	2010	2010	2013	2011
GUI	Web based	Python	Python + QT4	Y (initial phase)	Web based	N/A	Web based Java	N/A
Documentation	Good	Medium	Poor	Medium	Poor	Poor	Medium	Poor
Modularity	High	Poor	Poor	Poor	Poor	Poor	Medium	Medium
Distributed/ Centralized	D	C	C	C	C	C	C	C
TLS Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Platform Support	Linux Mac Windows	Linux	Linux Mac Windows	Linux	Linux Mac Windows	Linux Mac Windows	Linux Mac Windows	Linux
Open Stack Networking	Weak	Medium	No	Strong	Medium	Medium	Medium	Weak

IX. Conclusion

A study was conducted on SDN controllers to determine which eight controllers are now the most effective based on their level of deployment and use. The attributes of those eight controllers, including their modularity, productivity, and accessible interfaces, have been gathered via analysis of the controllers**.** After analyzing "Ryu" against our needs and the characteristics of the top eight controllers, we determined that "Ryu" is the most suitable option**.** The work that has been given may serve as a model for SDN developers or researchers to follow in order to assist ease the process of selecting an SDN controller**.** In the work that has to be done in the future, an ontology will need to be defined not only for the requirements of the business but also for the characteristics that will be offered by the controllers. This will ensure that no intermediary translation will be necessary for the matching process. In addition, it is possible to confirm the selection of the finest SDN controller by consulting knowledgeable SDN controller users. These users include specialists from businesses, data centers, and other types of facilities**.** In addition, feedback from users may be requested in order to determine the order in which the criteria are ranked.

Acknowledgments

The authors would like to thank the University of Baghdad and the Informatics Institute for Post Grad, for their help and encouragement in performing this work.

References

[1] Li, H., Li, P., Guo, S., & Nayak, A. (2014). Byzantine-resilient secure software-defined networks with multiple controllers in cloud. *IEEE Transactions on Cloud Computing*, *2*(4), 436-447.

[2] Waheed, S. R., Adnan, M. M., Suaib, N. M., & Rahim, M. S. M. (2020, April). Fuzzy logic controller for classroom air conditioner. *Journal of Physics: Conference Series* (Vol. 1484, No. 1, p. 012018). IOP Publishing.

[3] Salim, A. A., Ghoshal, S. K., Suan, L. P., Bidin, N., Hamzah, K., Duralim, M., & Bakhtiar, H. (2018). Liquid media regulated growth of cinnamon nanoparticles: Absorption and emission traits. *Malaysian Journal of Fundamental and Applied Sciences*, *14*(3-1), 447-449.

[4] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14-76.

[5] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T. V., & Kompella, R. (2013). Towards an elastic distributed SDN controller. *ACM SIGCOMM computer communication review*, *43*(4), 7-12.

[6] Khattak, Z. K., Awais, M., & Iqbal, A. (2014, December). Performance evaluation of OpenDaylight SDN controller. *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 671-676). IEEE.

[7] Salim, A. A., Bidin, N., Ghoshal, S. K., Islam, S., & Bakhtiar, H. (2018). Synthesis of truncated tetrahedral cinnamon nanoparticles in citric acid media via PLAL technique. *Materials Letters*, *217*, 267-270.

[8] Chekired, D. A., Khoukhi, L., & Mouftah, H. T. (2017). Decentralized cloud-SDN architecture in smart grid: A

dynamic pricing model. *IEEE Transactions on Industrial Informatics*, *14*(3), 1220-1231.

[9] Perrot, N., & Reynaud, T. (2016, March). Optimal placement of controllers in a resilient SDN architecture. *2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)* (pp. 145-151). IEEE.

[10] Raghunath, K., & Krishnan, P. (2018, July). Towards a secure SDN architecture. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-7). IEEE.

[11] Kleinrouweler, J. W., Cabrero, S., & Cesar, P. (2016, May). Delivering stable high-quality video: An SDN architecture with DASH assisting network elements. *Proceedings of the 7th International Conference on Multimedia Systems* (pp. 1-10).

[12] Salim, A. A., Bakhtiar, H., Krishnan, G., & Ghoshal, S. K. (2021). Nanosecond pulse laser-induced fabrication of gold and silver-integrated cinnamon shell structure: Tunable fluorescence dynamics and morphology. *Optics & Laser Technology*, *138*, 106834.

[13] Bakshi, K. (2013, March). Considerations for software defined networking (SDN): Approaches and use cases. *2013 IEEE Aerospace Conference* (pp. 1-9). IEEE.

[14] Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): A Survey. *Security and Communication Networks*, *9*(18), 5803-5833.

[15] Li, Y., & Li, J. (2014, November). MultiClassifier: A combination of DPI and ML for application-layer classification in SDN. *The 2014 2nd International Conference on Systems and Informatics* (ICSAI 2014) (pp. 682-686). IEEE.

[16] Waheed, S. R., Suaib, N. M., Rahim, M. S. M., Adnan, M. M., & Salim, A. A. (2021, April). Deep Learning Algorithms-based Object Detection and Localization Revisited. *In Journal of Physics: Conference Series (Vol. 1892*, No. 1, p. 012001). IOP Publishing.

[17] Rowshanrad, S., Abdi, V., & Keshtgari, M. (2016). Performance evaluation of SDN controllers: Floodlight and OpenDaylight. *IIUM Engineering Journal*, *17*(2), 47-57.

[18] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky. (2013). Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, ACM*. 1.

[19] M.P. Fernandez. (2013). Comparing openflow controller paradigms scalability: reactive and proactive. *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*. 1009-1016.

[20] Abbas, S. I., Hathot, S. F., Abbas, A. S., & Salim, A. A. (2021). Influence of Cu doping on structure, morphology and optical characteristics of SnO_2 thin films prepared by chemical bath deposition technique. *Optical Materials*, *117*, 111212.

[21] S.A. Shah, J. Faiz, M. Farooq, A. Shafi and S.A. Mehdi. (2013). An architectural evaluation of SDN controllers. Communications (ICC), *2013 IEEE International Conference on*. 3504-3508.

[22] D. Erickson. (2013). The beacon openflow controller. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, ACM, 2013*. 13-18.

[23] Salim, A. A., Ghoshal, S. K., & Bakhtiar, H. (2021). Growth mechanism and optical characteristics of Nd: YAG laser ablated amorphous cinnamon nanoparticles produced in ethanol: Influence of accumulative pulse irradiation time variation. *Photonics and Nanostructures-Fundamentals and Applications*, *43*, 100889.

[24] Khondoker, R., Zaalouk, A., Marx, R., & Bayarou, K. (2014, January). Feature-based comparison and selection of Software Defined Networking (SDN) controllers. *2014 World Congress on Computer Applications and Information Systems (WCCAIS)* (pp. 1-7). IEEE.

[25] Paliwal, M., Shrimankar, D., & Tembhurne, O. (2018). Controllers in SDN: A review report. *IEEE access*, *6*, 36256-36270.

[26] Salim, A. A., Bidin, N., & Islam, S. (2017). Low power CO_2 laser modified iron/nickel alloyed pure aluminum surface: Evaluation of structural and mechanical properties. *Surface and Coatings Technology*, *315*, 24-31.

[27] Hathot, S. F., Abbas, S. I., AlOgaili, H. A. T., & Salim, A. A. (2022). Influence of deposition time on absorption and electrical characteristics of ZnS thin films. *Optik*, *260*, 169056.

[28] Salim, A. A., Bakhtiar, H., Bidin, N., & Ghoshal, S. K. (2018). Antibacterial activity of decahedral cinnamon nanoparticles prepared in honey using PLAL technique. *Materials Letters*, *232*, 183-186.

[29] Wang, S. Y., Chiu, H. W., & Chou, C. L. (2015, April). Comparisons of SDN OpenFlow controllers over EstiNet: Ryu vs. NOX. *ICN* (Vol. 256).

[30] Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, Wu Chou. (2014). A roadmap for traffic engineering in SDN-OpenFlow networks. *Elsevier Computer Networks*, *71*, 1-30.

[31] Salim, A. A., Ghoshal, S. K., Shamsudin, M. S., Rosli, M. I., Aziz, M. S., Harun, S. W., ... & Bakhtiar, H. (2021). Absorption, fluorescence and sensing quality of Rose Bengal dye-encapsulated cinnamon nanoparticles. *Sensors and Actuators A: Physical*, *332*, 113055.

[32] Waheed, S. R., Rahim, M. S. M., Suaib, N. M., & Salim, A. A. (2023). CNN deep learning-based image to vector depiction. *Multimedia Tools and Applications*, 1-20.

[33] Hyojoon Kim, Nick Feamster. (2013). Improving network management with software defined networking. *IEEE Communications Magazine.*

[34] Salim, A. A., Bidin, N., Bakhtiar, H., Ghoshal, S. K., Al Azawi, M., & Krishnan, G. (2018, May). Optical and structure characterization of cinnamon nanoparticles synthesized by pulse laser ablation in liquid (PLAL). *Journal of Physics: Conference Series (Vol. 1027*, No. 1, p. 012002). IOP Publishing.

[35] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. (2014). A survey of software-defined networking: past, present, and future of programmable networks. Communications surveys and tutorials. *IEEE Communications Society, Institute of Electrical and Electronics Engineers, 16*(3), 1617-16342.

[36] Andri, C., Alkawaz, M. H., & Waheed, S. R. (2019, June). Examining effectiveness and user experiences in 3d mobile based augmented reality for msu virtual tour. *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)* (pp. 161-167). IEEE.

[37] Kaur, S., Singh, J., & Ghumman, N. S. (2014, August). Network programmability using POX controller. *ICCCS International Conference on communication, Computing & Systems, IEEE (Vol. 138*, p. 70). sn.

[38] Waheed, S. R., Saadi, S. M., Rahim, M. S. M., Suaib, N. M., Najjar, F. H., Adnan, M. M., & Salim, A. A. (2023). Melanoma skin cancer classification based on CNN deep learning algorithms. *Malaysian Journal of Fundamental and Applied Sciences*, *19*(3), 299-305.

[39] Islam, M., Islam, N., & Refat, M. (2020). Node to node performance evaluation through RYU SDN controller. *Wireless Personal Communications*, *112*(1), 555-570.

[40] Aldhuhaibat, M. J., Amana, M. S., Aboud, H., & Salim, A. A. (2022). Radiation attenuation capacity improvement of various oxides via high density polyethylene composite reinforcement. *Ceramics International*, *48*(17), 25011-25019.

[41] Alluhaybi, H. A., Ghoshal, S. K., Shamsuri, W. W., Alsobhi, B. O., Salim, A. A., & Krishnan, G. (2019). Pulsed laser ablation in liquid assisted growth of gold nanoparticles: Evaluation of structural and optical features. *Nano-Structures & Nano-Objects*, *19*, 100355.

[42] Cai, Z., Cox, A. L., & Ng, T. S. (2010). *Maestro: A system for scalable openflow control*.

[43] Kadhim, K. A., Najjar, F. H., Waad, A. A., Al-Kharsan, I. H., Khudhair, Z. N., & Salim, A. A. (2023). Leukemia classification using a convolutional neural network of aml images. *Malaysian Journal of Fundamental and Applied Sciences*, *19*(3), 306-312.

[44] Waheed, S. R., Sakran, A. A., Rahim, M. S. M., Suaib, N. M., Najjar, F. H., Kadhim, K. A., Salim, A. A. & Adnan, M. M. (2023). Design a crime detection system based fog computing.