EasyChair Preprint

№ 10068

Network Management Devices in an SDN Environment

Diyar Jamal Hamad, Khirota Gorgees Yalda, Tapus Nicolae and Ibrahim Taner Okumus

May 10, 2023

Network Management devices in an SDN environment

Diyar Jamal Hamad
Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest
Bucharest, Romania
Soran Technical college
Erbil Polytechnic University
Iraq, Erbil
diyar.hamad@epu.edu.iq

Khirota Gorgees Yalda
Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest
Bucharest, Romania
Soran Technical college
Erbil Polytechnic University
Iraq, Erbil
kherota.yalda@epu.edu.iq

Nicolae Tapus
Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest
Bucharest, Romania
nicolae.tapus@upb.ro

Ibrahim Taner Okumus
Computer Engineering
Kahramanmaras Sutcu Imam University
Kahramanmaras, Turkey
iokumus@ksu.edu.tr

***Abstract*— Network optimization and continued availability depend on a number of capabilities that are part of network management. The Maintenance, operating and also offering a safeguarded interaction network is very complicated. It calls for the network operators to grapple with low-level vendor particular arrangements to execute the high degree network policies which are complicated. A method for monitoring networks with OpenFlow controller is presented in this paper in two separate functions for the same networks. Bandwidth utilization, Meter values, charts and statistics are provided by the method to extend controller monitoring capabilities. The method architecture and implementation will be introduced in order to present the feature set. Additionally, softswitches are used as a switch and Mininet to evaluate a virtualized network. This analysis shows whether Meters or ports value is better for network management.**

Keywords— SDN, Network Management, Traffic engineering, Control Plane, and openflow.

Introduction

SDN involves separating the data plane from the control plane. Flow tables keep forwarding rules associated with forwarding rules stored on data plane devices (switches and routers). Each device's flow table is managed by the control plane, which runs on a different device. Device configuration can be done dynamically, and a global network view can be obtained. A new network pattern seeks to remedy the flaws of the existing network infrastructures by removing the vertical integration paradigm. Despite being managed by a centralized logical controller inside a network operating system, it as well separates network control logic apart from routers and switches, as illustrated in Figure 1 which shows how the SDN architecture integrates the control logic from the forwarding hardware and enables simpler decision-making, middlebox consolidation, and the consolidation of new additional functionality. The relationships between the data planes described by solid lines and the controlling plane's ties by red lines [1]. Actual, centralized management layers will enable a more adaptable and effective strategy. The ideal example of OpenFlow will segregate the power and data planes via an application programming interface (API) between the SDN controller and switches [2].

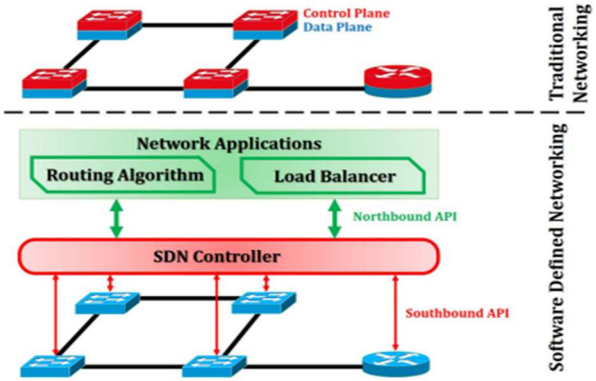


Fig. 1. compares SDN and conventional networking designs [3].

Software-defined networking is one of the most extensively studied fields due to the rapidly expanding market and adherence to end-to-end communication protocols. It is a networking model that incorporates a variety of abilities and resolves the problems with previous networking models. A centralized controller is integrated into the software defined network to separate network intelligence and packet swapping hardware. Then, using Free Flow protocols installed at switches, this controller acts as the major controller or the primary brain in charge of determining routing. This method, for which the intelligent network layer is referred to as the control layer, can be regarded as one of the advantageous methods [4]. For a variety of reasons, the data layer and control layer are separated [5]. This model offers excellent consistency and permits increased network abstraction, simpler network management, and innovation potential [2], allowing the controller—an external body—to be presented as the control layer [6]. The data plane remains in packet forwarding components called switches [7]. The packet-forwarding switches still contain the data plane. The controller and networking elements (switches) are coordinated using the free flow protocol. As information is transported from networking elements and gathered in the outer controller, the switch cost reduces. Since the central controller is fully knowledgeable about the network, managing and using it is simple [8]. No matter what network infrastructure is used to connect devices from different manufacturers, the SDN

architecture enables uniform administration of the data route components. All intelligence is combined under unified management, which also maintains a network-wide understanding of the components and relationships that bind the data path's components. The Network Operating System (NOS) is excellent for network management (NM) tasks due to this centralized, modern viewpoint [9].

An SDN-based intra-domain routing and resource management model is presented in this study. A pre-established multi path (PMP) allows scalable network management by virtualizing the underlying network. Paths between ingress-egress pairs are predefined. Routing does not take core switches into account along the paths. According to these paths, the controller manages intradomain routing and resources.

I. Related Work

A couple of the relevant studies mentioned in this part are used as references in [10] [11] [12] presented four network management and control dimensions. They are data, decision, dissemination, and discovery. In our study, we have applied this basic way of looking at network administration and control. In the reference [13]has emphasized the necessity of independence in managing networks for next-generation networks. These pieces of work served as our point of reference for future study into the application of independence and the necessity of dynamic and automatic actions for software defined networks. T. Feng et. al. [14] presented a method of dynamic traffic isolation using the principles of software defined networking. This study has established a software defined networking expectation area from network management systems. A software-defined networking-oriented architecture has been suggested for network operating systems by J. Mueller et. al. [15], that processes packets, manages open devices, and understands network state cognitively. This study has been cited in order to comprehend the future directions of research in the extending fields of network management and operating systems. A new method for traffic engineering in software defined networking was suggested by [16]. This work was cited by us in order to comprehend how network management may dynamically adjust to altering network configuration behavior. [17] had explored employing high level language for network configuration, activating a network for rapid alters, and offering network visualization while debugging network problems. We now have a better understanding of what to expect from software defined networking management thanks to this study. The literature contains a number of routing and resource management focused SDN and non-SDN based works. In order to handle tunnels in MPLS networks, a server-based paradigm has been developed by [18]. In this system, the server has access to network-wide data. The crucial and shared linkages between the ingress-egress couple are determined. A weighted graph where the expense of vital linkages being high is built and continually updated. To achieve improved QoS, routing across Label Switching Paths (LSPs) that utilize crucial links is avoided. Through load balancing between several channels linking ingress-egress couple, intended to prevent path congestion in MPLS networks. There are various ideas for centralized management of traffic engineering [19]. Data centers can balance data traffic using the flow scheduling technique called Hedera. It catches oversized flows during the time and the global first fit algorithm searches for appropriate routes on the tree. There is signaling overhead because controllers must add new rules to the switches along the pathways. A distributed architecture for enterprise networks is called DIFANE [20]. The most important network performance metrics are bandwidth usage, network latency, packet loss rates and throughput rates. Van Adrichem et al. [21] Suggests using the OpenNetMon network monitoring module, which develops a number of techniques for measuring network performance parameters. The OpenNetMon can continuously track network delays, throughput, and packet loss rates. In response to messages the SDN controller receives, FlowSense creates a monitoring module for the controller that can assess dynamic alters in network flows. For instance, after receiving a FlowRemoved message for a specific flow, the controller divides the statistical values of the flow by the size of the associated flow to get the flow's throughput rate [22].

II. OpenFlow

With OpenFlow, software switches contain two parts: a data plane and a control plane. Classical networking devices have a forwarding plane (data plane) and a routing plane which works like a control plane. Controlling different devices is a skill that network administrators must learn. There are several companies that sell those devices, and every company has its own rules. There is a lot of difficulty and inconvenience involved. The OpenFlow protocol gives network administrators a new way to manage their networks. Routing rules are controlled by the controller, and packet forwarding is handled by the hardware.

III. Traffic Engineering

Managing networks aims to increase network performance and maintain network availability. Improved quality of service can be achieved by scheduling network traffic appropriately. With SDN, traffic can be scheduled between sources and destinations via multiple paths.

Through real-time traffic analysis, traffic prediction, and routing design, Traffic Engineering (TE) plays a critical role in optimization of network performance [23].

A SDN can be used to manage traffic flows, provide fault tolerance, update topologies, analyze and characterize traffic, which are divided into the following sections in Figure 2. In the control and data planes, flow management addresses traffic overhead issues. It refers to the ability of a network component (such as a controller, switch, or link) to recover quickly from a fault [24].

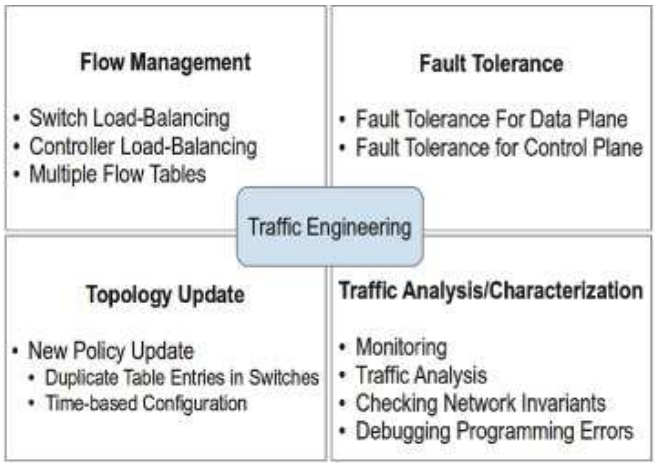


Fig. 2. Traffic engineering activities [24]

In general, IP-based TE solves the problem of multipath traffic load balance by optimizing the IP routing algorithm to avoid network congestion [25]. For example, in the reference [26] propose a neighborhood search algorithm, which is based on link weights of Open Short Path First (OSPF) to adjust the routing calculation strategy, and finally get multiple equivalent shortest paths to achieve traffic load balance. Chen et al. [27] Develop a multipath planning approach for IoT multimedia sensing.

IV. MEASURING IN SDN

In legacy networks, each measurement approach needs a detachment of hardware installation or software setup. This makes using the approach a time-consuming and costly task. OpenFlow networks, on the other hand, offer the required interfaces to execute the majority of the conventional measurement techniques at a cheaper cost and with more efficiency [28]. The OpenFlow protocol provides two types of messages that are used for collecting statistics for this work [29]. A statistics request message and statistics reply message make up the two OpenFlow messages. A message asking for the switch's current ports and flow statistics is known as a statistics request message sent from the controller to the switch. The switch's response to the controller's request is the message known as the statistics reply message. Network measurements are frequently made at the network or application plane [30]. Measurements of the application plane are intended to measure the effectiveness of the application. Infrastructure forwarding elements are intended to be used in network plane measurements (such as routers and switches) [31]. The purpose of this work is to gather accurate data on network traffic that may be utilized to confirm any changes in network performance that may occur.

V. DESIGN AND IMPLEMENTATION MODULES FOR MANAGEMENT

To manage an SDN network, collecting traffic statistics is helpful. Our idea is aimed to extract specific types of statistics from the generated traffic in the network. Based on Dijkstra Algorithm which we changed from shortest path to the widest path, we have created two separated Modules in Floodlight controller and it is specially designed to collect different statistics from switches, Java programming and eclipse is used for developed modules based on floodlight controller which is created by Java. The design and the implementation of this module are described in the next two paragraphs.

A. Design

Since OpenFlow is designed to store flow entries in the switch flow tables, the switch flow tables are then able to forward flow entries to their destinations based on the instructions that are received from the OpenFlow controller. Switch flow tables have a number of entries, one of which is counters. It is important to note that the counters entry contains both packet counters and byte counters. These two counters in SDN are used to store all the packets and bytes received during the course of the protocol. Our modules are designed to extract the actual numbers stored in these two counters and find bandwidth in every 10 seconds. To compare with other solutions, we used the same time slot.

B. Implementation

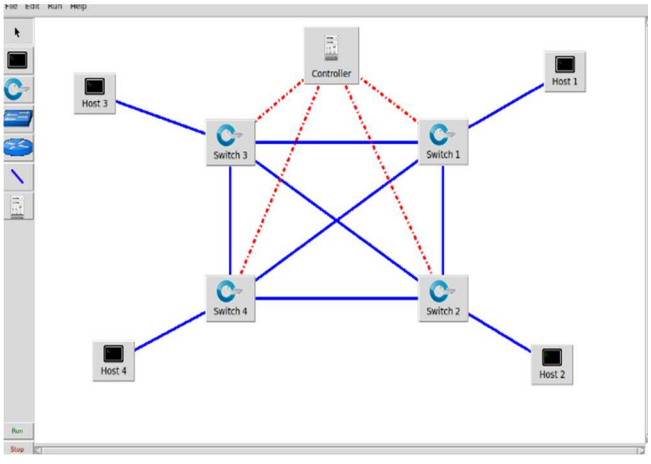
In order to create the presented design, Java programming language is used. The reason behind using Java is that Floodlight controller is Java-based. In both modules, statistics collecting functions are implemented on floodlight controller in order to query the packet and byte counters periodically. The numbers amassed in those counters, in addition to the variety of flows, might be displayed at the CLI in a brief length of time. The interval time between each appearance and the next for the statistics is ten seconds. The collected statistics is imported to an excel sheet and it's used to analyze traffic to show in-depth detail. This list shows the type of protocol used to generate the traffic, source IP address, destination IP address and some extra information about segmenting. Mainly, this Floodlight controller module is implemented to measure two different types of traffic statistics. The first one is to find bandwidth in each port, the second one is to find statistics from switches directly. Those modules are collecting statistics provided by OpenFlow request such us:

- STATISTICS REQUEST: A controller sends a message to a switch asking for its current statistics on flows, ports, etc.
- STATISTICS REPLY: Responds to a request message sent by the controller from the switch.

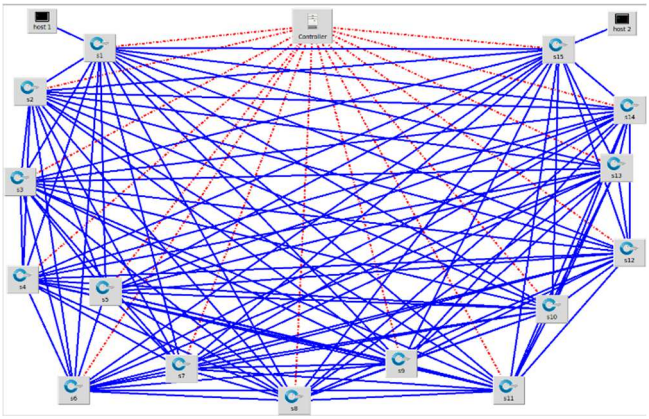
This process uses the application layer in SDN. This module can also be used to measure TCP and ICMP requests that use the network layer. As such, our Floodlight controller modules can be thought of as multifunctional pieces of equipment.

VI. EXPERIMENT AND RESULTS

Mininet can be used to implement SDN-capable network topologies. A Linux kernel can run end-hosts, switches, routers, and application code on a realistic virtual network created with Mininet [32]. In seconds, Mininet can launch switches, controllers, and hosts via a virtual machine called Mininet VM. As part of our SDN environment, we create our custom topology using MiniEdit. This experiment uses four end-hosts, four ofsoftswitch [33] and a Floodlight [34] controller as shown in Figure 3.a.



a. Network with four switches



b. Network with fifteen switches

Fig. 3. Topologies

The experiment used Floodlight as a network controller for OpenFlow and for our test environments we used the Iperf [35]tool to manage traffic with 5Mb from first host to last host or in the second scenario we manage a 5MB traffic from first which linked to first switch to second host which linked with switch fifteen. With the introduction of SDN, many controllers have been developed [1]. However, floodlight is a popular controller. With Floodlight, OpenFlow switches can be used both physically and virtually, depending on the configuration you choose. It is Java-based and based on the Beacon controller implementation developed at Stanford University. Since the floodlight controller imported to a Java application such as Eclipse, then our Modules are implemented and run the controller, then the topology is creating as a custom topology because our SDN network is connected complexly, and controller works by directly connecting the OpenFlow switch to the network to receive instructions or modifications. Messages OFPT_STATS_REQUEST and OFPT_STATS_REPLY are used to read state messages in SDN structures [36]. Datapaths can be queried about their current state using OFPT_STATS_REQUEST messages, while switches respond with OFPT_STATS_REPLY messages. Specifying the type of information in a request message or response determines what the body field means.

With OpenFlow, switch statistics can be obtained at the port level in SDN. The OpenFlow switch generates the "OFPortStatsReply" message in response to an "OFPortStatsRequest" message. It is possible to collect more information about packets generated in a network by using these two messages. It can either request statistics on a specific port, if the port_no field contains the port number, or for all ports, if the port_no field contains "OFPP_ANY". When the controller queries the ports iteratively, it receives updates to the statistics counters which contain numbers of packets and bytes for each port used. The statistics of the ports can provide more information about both the send and receive states, such as errors, collisions, dropped packets and dropped bytes, which can be used to calculate the path loss rate [37]. Numbers of received packets, transmitted packets, received bytes, and transmitted bytes are included in the reply body [38]. Furthermore, we started our work by managing traffic with 5Mb to the whole network hosts, and our module could collect the port statistics from each switch in the network and we found Bandwidth in each link via the Port Statistic in each 10 seconds as presented in figure 4.

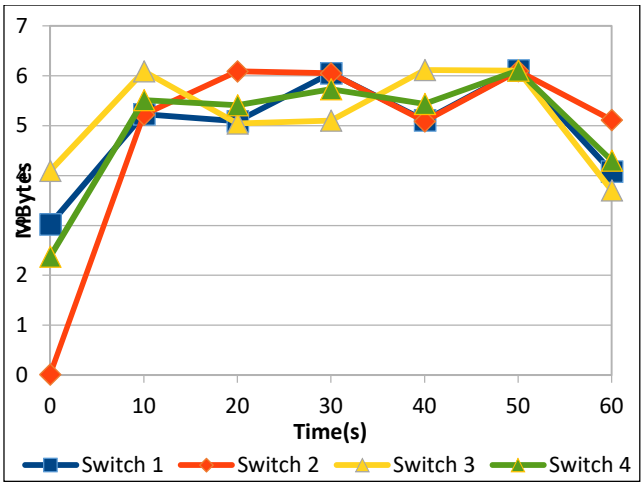


Fig. 4. Port Statistics from switches.

The second way to manage or control an SDN network is finding another statistic from a new functionality which works with OpenFlow version 1.3 or higher protocols. The function is Meter stats which we implemented to our controller to get Meter Table values and find the whole information which included in Meter Table such as packet_count, byte_count, meterId, flow_count, packet_in_count, byte_in_count and duration. The packet_count and byte_count is up to meters and the packet_in_count and byte_in_count is turned out for Flow because meters are working with Flow and before getting the Meter values, we made flows for the whole links with switch otherwise the value shown as zero. The collected statistic from Meters value is shown in Figure 5.

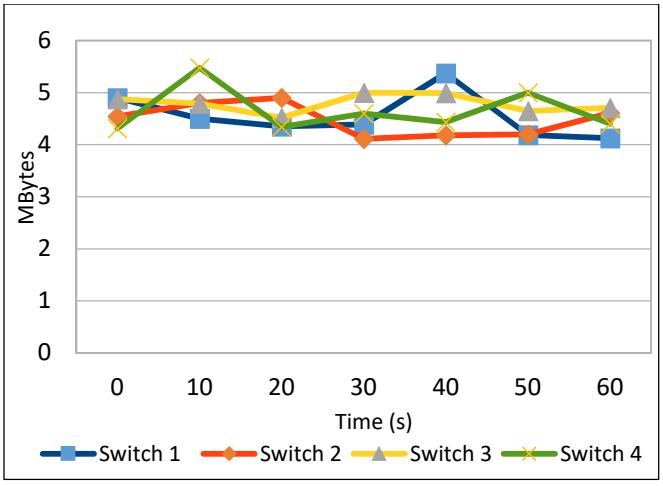


Fig. 5. Meter Statistics from switches.

Based on Dijkstra both Port and Meter values are helping us to manage or control the network, for example at figure 4 at the second 30, third switch has lowest bandwidth so the Dijkstra uses this switch to send the packets through and we can see at second 40 the same switch has higher bandwidth, but in figure 5 the controller has meter values so it can manage the network better than based on port values, because we can see that controller getting values of each switch then it is deciding which path should be use, and in both figure (4 and 5) we can see that the values sometimes is bigger than traffic, it is because the controller has background traffic and sending messages in each 10 seconds to make sure of the network, at the same time there is no guarantee to make a real traffic as

shown in virtuality and we can see that background traffic was not affected on meters as affected with ports.

We performed the same actions on topology B in figure 3 and obtained the same results as topology A in the same figure. We did not show the results in this paper because they were a little complicated and hard to understand. We used fifteen switches and if we wanted to get the statistics of each of the switches, the result of this work would be a graph with fifteen lines that were very close to each other.

CONCLUSION

A new implementation for measuring traffic in software-defined networks has been presented in this paper. It is a concise list of the most important statistics in the network (packets and bytes). Our modules have the capability to measure UDP traffic which is generated by the Iperf tool, and we could get two separate values from Port and Meter, after calculating and finding the bandwidth and the packet routing path, we obtained that managing an SDN network by choosing the meter value is better than managing by port statistics. It is important to view what Wireshark displays to ensure our module can measure traffic generated by different protocols simultaneously.

ACKNOWLEDGMENT

This work was supported by a grant of the Ministry of Research, Innovation and Digitalization, project CloudPRECIS, MySmis Code: 124812, within POC.

REFERENCES

[1] B. A. 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, doi: 10.1109/SURV.2014.012214.00180.

[2] T. G. Thajeel and A. Abdulhassan, "A Comprehensive Survey on Software-Defined Networking Load Balancers," in *2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA)*, IEEE, Sep. 2021, pp. 1–7. doi: 10.1109/IICETA51758.2021.9717919.

[3] B. R. Al-Kaseem and H. S. Al-Raweshidy, "SD-NFV as an Energy Efficient Approach for M2M Networks Using Cloud-Based 6LoWPAN Testbed," *IEEE Internet Things J*, vol. 4, no. 5, pp. 1787–1797, Oct. 2017, doi: 10.1109/JIOT.2017.2704921.

[4] M. Hamdan *et al.*, "A comprehensive survey of load balancing techniques in software-defined network," *Journal of Network and Computer Applications*, vol. 174, p. 102856, Jan. 2021, doi: 10.1016/j.jnca.2020.102856.

[5] M. A. R. AlShehri and S. Mishra, "Feature Based Comparison and Selection of SDN Controller," *International Journal of Innovation and Technology Management*, vol. 16, no. 05, Aug. 2019, doi: 10.1142/S0219877019500299.

[6] R. Santos, H. Ogawa, G. K. Tran, K. Sakaguchi, and A. Kassler, "Turning the Knobs on OpenFlow-Based Resiliency in mmWave Small Cell Meshed Networks," in *2017 IEEE Globecom Workshops (GC Wkshps)*, IEEE, Dec. 2017, pp. 1–5. doi: 10.1109/GLOCOMW.2017.8269214.

[7] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G Networks: Use Cases and Technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, Mar. 2020, doi: 10.1109/MCOM.001.1900411.

[8] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network Monitoring in Software-Defined Networking: A Review," *IEEE Syst J*, vol. 12, no. 4, pp. 3958–3969, Dec. 2018, doi: 10.1109/JSYST.2018.2798060.

[9] A. Devlic, W. John, and P. Skoldstrom, "A Use-Case Based Analysis of Network Management Functions in the ONF SDN Model," in *2012 European Workshop on Software Defined Networking*, IEEE, Oct. 2012, pp. 85–90. doi: 10.1109/EWSDN.2012.11.

[10] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, Oct. 2005, doi: 10.1145/1096536.1096541.

[11] N. Samaan and A. Karmouch, "Towards Autonomic Network Management: an Analysis of Current and Future Research Directions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 22–36, 2009, doi: 10.1109/SURV.2009.090303.

[12] L. Ciavaglia, "UniverSelf, Realizing Autonomics for Future Networks," 2013, pp. 363–366. doi: 10.1007/978-3-642-38082-2_36.

[13] A. Lara, A. Kolasani, and B. Ramamurthy, "Simplifying network management using Software Defined Networking and OpenFlow," in *2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS)*, IEEE, Dec. 2012, pp. 24–29. doi: 10.1109/ANTS.2012.6524222.

[14] T. Feng, J. Bi, and H. Hu, "TUNOS: A novel SDN-oriented networking operating system," in *2012 20th IEEE International Conference on Network Protocols (ICNP)*, IEEE, Oct. 2012, pp. 1–2. doi: 10.1109/ICNP.2012.6459936.

[15] J. Mueller, A. Wierz, and T. Magedanz, "Scalable On-Demand Network Management Module for Software Defined Telecommunication Networks," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, IEEE, Nov. 2013, pp. 1–6. doi: 10.1109/SDN4FNS.2013.6702550.

[16] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, Feb. 2013, doi: 10.1109/MCOM.2013.6461195.

[17] I. Aijaz and S. Idrees, "Performance Evaluation of Multi-protocol Label Switching-Traffic Engineering Schemes," *ICST Transactions on Mobile Communications and Applications*, vol. 6, no. 19, p. 166550, Jul. 2021, doi: 10.4108/eai.8-10-2020.166550.

[18] K.-F. Hsu, P. Tammana, R. Beckett, A. Chen, J. Rexford, and D. Walker, "Adaptive Weighted Traffic Splitting in Programmable Data Planes," in *Proceedings of the Symposium on SDN Research*, New York, NY, USA: ACM, Mar. 2020, pp. 103–109. doi: 10.1145/3373360.3380841.

[19] M. Fraga, M. Micheletto, A. Llinás, R. Santos, and P. Zabala, "Flow Scheduling in Data Center Networks with Time and Energy Constraints: A Software-Defined Network Approach," *Future Internet*, vol. 14, no. 2, p. 65, Feb. 2022, doi: 10.3390/fi14020065.

[20] D. B. Rawat and S. R. Reddy, "Software Defined Networking Architecture, Security and Energy Efficiency: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 325–346, 2017, doi: 10.1109/COMST.2016.2618874.

[21] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, IEEE, May 2014, pp. 1–8. doi: 10.1109/NOMS.2014.6838228.

[22] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," 2013, pp. 31–41. doi: 10.1007/978-3-642-36516-4_4.

[23] D. K. Dake, J. D. Gadze, G. S. Klogo, and H. Nunoo-Mensah, "Traffic Engineering in Software-defined Networks using Reinforcement Learning: A Review," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, 2021, doi: 10.14569/IJACSA.2021.0120541.

[24] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, Oct. 2014, doi: 10.1016/j.comnet.2014.06.002.

[25] T. Rahman, I. Ahmad, A. Zeb, I. Khan, G. Ali, and M. ElAffendi, "Performance Evaluation of Routing Protocols for Underwater Wireless Sensor Networks," *J Mar Sci Eng*, vol. 11, no. 1, p. 38, Dec. 2022, doi: 10.3390/jmse11010038.

[26] S. Bing, Y. Haiyang, L. Jieru, and M. Zhenghua, "Traffic Optimization on the Dynamic Switching of ABR for OSPF Networks," in *2009 International Conference on Information Technology and Computer Science*, IEEE, Jul. 2009, pp. 429–432. doi: 10.1109/ITCS.2009.226.

[27] M. Chen *et al.*, "M-plan: Multipath Planning based transmissions for IoT multimedia sensing," in *2016 International Wireless*

Communications and Mobile Computing Conference (IWCMC), IEEE, Sep. 2016, pp. 339–344. doi: 10.1109/IWCMC.2016.7577081.

[28] H. Yahyaoui, M. F. Zhani, O. Bouachir, and M. Aloqaily, "On minimizing flow monitoring costs in large-scale software-defined network networks," *International Journal of Network Management*, vol. 33, no. 2, Mar. 2023, doi: 10.1002/nem.2220.

[29] J. Matousek, A. Lucansky, D. Janecek, J. Sabo, J. Korenek, and G. Antichi, "ClassBench-ng: Benchmarking Packet Classification Algorithms in the OpenFlow Era," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 1912–1925, Oct. 2022, doi: 10.1109/TNET.2022.3155708.

[30] L. Yang, B. Ng, W. K. G. Seah, L. Groves, and D. Singh, "A survey on network forwarding in Software-Defined Networking," *Journal of Network and Computer Applications*, vol. 176, p. 102947, Feb. 2021, doi: 10.1016/j.jnca.2020.102947.

[31] Dandan Zou, Jianbing Ding, Xidong Wang, Xiaozhou Ye, and Ye Ouyang, "Research on network cloud equipment anomaly and root cause analysis," *ITU Journal on Future and Evolving Technologies*, vol. 3, no. 2, pp. 89–97, May 2022, doi: 10.52953/TVLO2995.

[32] O. Romanov, I. Saychenko, A. Marinov, and S. Skolets, "RESEARCH OF SDN NETWORK PERFORMANCE PARAMETERS USING MININET NETWORK EMULATOR," *Information and Telecommunication Sciences*, no. 1, pp. 24–32, Jun. 2021, doi: 10.20535/2411-2976.12021.24-32.

[33] N. Bonelli, G. Procissi, D. Sanvito, and R. Bifulco, "The acceleration of OfSoftSwitch," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, Nov. 2017, pp. 1–6. doi: 10.1109/NFV-SDN.2017.8169842.

[34] 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, Nov. 2016, doi: 10.31436/iiumej.v17i2.615.

[35] Dr. J. D. Gadze, K. A. Obeng, and J. Owusu-Agyeman, "Dynamic Bandwidth Utilization in Software - Defined Campus Based Networks: A Case Study of the Kwame Nkrumah University of Science and Technology," *IJARCCE*, vol. 9, no. 7, pp. 94–109, Jul. 2020, doi: 10.17148/IJARCCE.2020.9617.

[36] M. Xiao, Y. Cui, Q. Qian, and G. Shen, "KIND: A Novel Image-Mutual-Information-Based Decision Fusion Method for Saturation Attack Detection in SD-IoT," *IEEE Internet Things J*, vol. 9, no. 23, pp. 23750–23771, Dec. 2022, doi: 10.1109/JIOT.2022.3190269.

[37] P. Göransson, C. Black, and T. Culver, "The OpenFlow Specification," in *Software Defined Networks*, Elsevier, 2017, pp. 89–136. doi: 10.1016/B978-0-12-804555-8.00005-3.

[38] M. O. Elbasheer, A. Aldegheishem, J. Lloret, and N. Alrajeh, "A QoS-Based routing algorithm over software defined networks," *Journal of Network and Computer Applications*, vol. 194, p. 103215, Nov. 2021, doi: 10.1016/j.jnca.2021.103215.