Comprehensive Exploration and Performance Assessment of SDN Controller

Fadhl E. Hadi[1, a)], Bilal R. Al-Kaseem[2, b)], Hamed S. Al-Raweshidy[3, c)]

[1] Department of Computer Engineering, College of Engineering, Al-Iraqia University, Baghdad, Iraq
[2] Department of Communication Engineering, College of Engineering and Information Technology, AlShaab University, Baghdad, Iraq
[3] Department of Electronic and Computer Engineering, College of Engineering, Design and Physical Sciences, Brunel University London, London, United Kingdom

a) Corresponding author: fadhl.e.hadi@aliraqia.edu.iq
b) bilal.al-kaseem@alshaab.edu.iq
c) hamed.al-raweshidy@brunel.ac.uk

Abstract

SDNs provide flexible and intelligent network capabilities by dividing traditional networks into a centralized control plane and a programmable data plane. Network performance improvement relies on the intelligent control plane, which establishes flow paths for switches. Within the control plane, the controller serves as the core component for all data plane management operations, underscoring the significance of its performance and capabilities. Additionally, it is crucial to utilize accurate and efficient tools for assessing various evaluation parameters. Specifically, based on their capabilities, we identify and categorize 14 controllers, offering a qualitative comparison of their features. Furthermore, we investigate the capabilities of benchmarking tools used to assess controllers for SDN.

Keywords- SDN Controllers, Performance, Software Defined Network.

INTRODUCTION

In recent times, SDN has seen substantial expansion and deployment across diverse network types. This includes data center networks, which have actively embraced SDN [1], as well as wireless networks, Internet of Things networks [2], cellular networks and wide area [3], and domains related to security and privacy [4]. SDN, as a network architecture, fundamentally separates control logic from network devices. This separation empowers centralized traffic management and flow control. The SDN architecture comprises multiple layers: the control plane, the data plane, and the management plane. Controllers in the control plane program devices are responsible for data forwarding in the data plane, while the management plane works with the control layer to create and enforce policies throughout the whole network.

Traditional networks face various limitations in areas such as flow management, traffic engineering, policy enforcement, virtualization, and security. These limits stem from a variety of service requirements as well as network scale [5]. By divorcing data traffic forwarding from network control intelligence, SDN provides an efficient and streamlined solution to these difficulties. Consequently, network switches become simplified forwarding devices that follow instructions from a software-based controller. The centralized nature of SDN allows for programmatic control over the entire network and allows for real-time administration of underlying devices. SDN implementation simplifies network administration and removes the constraints associated with rigid network structures. Notable SDN controllers include NOX [6], POX [7], Floodlight [8], OpenDaylight [9], Open Network Operating System [10], and RYU [11]. However, it's worth noting that various controllers and their variants have been documented in the literature. In practice, selecting the most suitable controller for a particular network type can be a challenging task.

ARCHITECTURE OF SDN

SDN is a new technology that revolves around the concept of collecting intelligence from network devices. Employing a centralized controller to oversee and govern all network operations. Figure 1 provides an illustration of the core architecture of SDN, in which The application layer, control layer, and infrastructure layer are the three separate layers.

1. Infrastructure Layer: The network layer of SDN is made up of numerous network devices, Routers, switches, and access points are examples of networking devices. Within this layer, Both virtual switches (such as Open

vSwitch, Indigo, Pica8, Nettle, and Open Flowj) and physical switches (as described in [12]) are supported. play crucial roles. The data plane's principal job is to forward packets depending on predefined rules and regulations.

2. Control Layer: Positioned between the application layer and the infrastructure layer, the control plane encompasses a controller that governs the overall operations of the SDN. This layer acts as an intermediary, responsible for regulating traffic flow and making routing, flow forwarding, and packet dropping decisions using programmable methods [13]. In distributed environments, controllers establish communication among themselves utilizing east- and west-bound interfaces. Furthermore, South-bound APIs such as OpenFlow and NetConf are used to communicate between the control layer and the infrastructure layer, facilitating seamless interaction and coordination [14].

3. Application Layer: SDN's application layer is accountable for handling software-related functions and safety applications. It encompasses various applications like virtualisation of networks, IDS are intrusion detection systems, intrusion detection and prevention systems (IPS), Implementation of a firewall, as well as mobility management. This layer establishes communication with the control layer through the (A-CPI), also known as the application interface on the northbound side [15] as shown in Fig. 1.

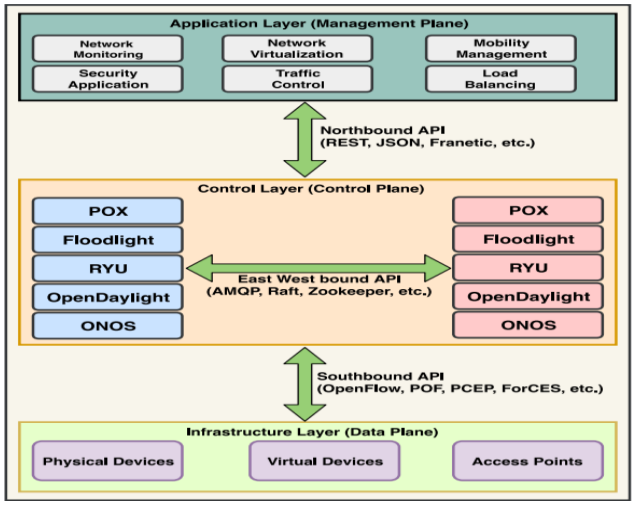

Fig. 1: layered structure of SDN and their Components.

I. SDN CONTROLLERS

The controller holds a central role within any SDN setup, offering a comprehensive a bird's-eye view of the entire network, encompassing both the information plane and SDN devices. It serves as the bridge connecting these resurrected resources with management applications and executes flow of events as directed by means of application policies across these devicesThis section contains, we introduce the standard control architecture and trace their development to contemporary controllers. Additionally, we provide a categorization and comparative analysis of 14 distinct controllers.

A. Architecture of SDN Controllers

Within a network that is software-defined, commonly known as (NOS), the controller serves as the pivotal and essential element tasked with decision-making regarding network management.

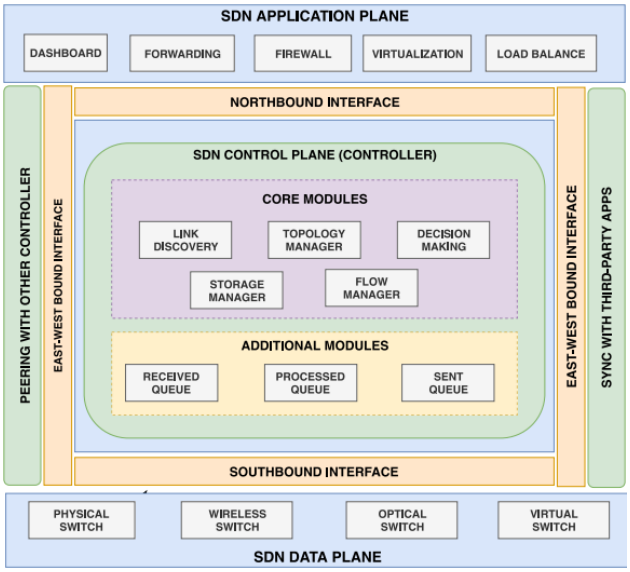

Fig. 2: General Overview of SDN Controller

The management of traffic within the beneath the surface network is a key concern. Various proposals in the literature do not fundamentally alter the core architecture of the controller; instead, they exhibit variations in terms of modules and capabilities. Consequently, presenting detailed individual architectures may not offer significant value to the reader. In this context, we provide a broad overview of the overall structure, illustrated in Figure 2, and proceed to examine its distinct modules.

Controller Core: The primary responsibilities the controller is closely tied to managing the network's topology and the flow of traffic. The link finding module on a regular basis sends queries through external connectors using messages sent in packets. These queries are answered in the form of message packet, enabling the controller construct the network's topology. The topology in and of itself is supervised via way of the topology manager, facilitating utilizing the topology manager in determining the most efficient paths between network nodes. These paths are constructed in a manner that allows for the enforcement of various QoS policies or security measures during path establishment. Furthermore, the controller may include specialized modules like a collector/manager of statistics and a queue supervisor. These serve the purposes of gathering performance data and overseeing There are separate incoming and outgoing packet queues, respectively. The flow controller stands out as one of the key modules, directly engaging with the Flow entries and flow tables in the data plane, all made possible through the utilization of the southbound interface.

Interfaces: The central controller is encircled by various interfaces designed for interaction with several network layers and devices. The (SBI) specifies a set of rules that facilitate Forwarding of packets between forwarding devices and controllers. SBI plays a crucial role in enabling intelligent provisioning of Networking devices, both physical and virtual. (OF) [16] is the prevailing SBI, widely adopted as an industry standard. Its primary responsibility is to design flows and categorize network traffic based on predetermined criteria rules.

Conversely, the controller employs the (NBI) to enable application developers to integrate their applications with both the Devices for the controller and data plane. While controllers assist various APIs heading north, many are built on REST APIs. For communication between controllers, the West Bound Interface (WBI) is utilized. It's worth noting that there isn't a standardized This requires a communication interface., resulting in various controllers relying on various mechanisms. Furthermore controllers that are heterogeneous typically they do not communicate with one another.

The (EBI increases the controller's capabilities to communicate with outdated routers. BGP [16] serves as most frequently employed This protocol is intended for this specific purpose.

B. **Evolution of SDN Controllers**

The design of modern SDN controllers of SDN networks represent a departure from the initial efforts to centralize network control, which date back to the mid-2000s. During that time, several attempts were created to segregate control logic from the data plane. For instance, SoftRouter [17], [18] and ForCES [18] were presented as solitary network devices designed to separate (CEs) from (FEs). However, these solutions were only capable of modifying packets functions because the majority of routers at that time lacked the intelligence in computing and network comprehension needed to carry out more advanced operations. (RCP) [19] were offered as intra-AS (Autonomous System). platforms to create a BGP control platform that can be expanded. Nonetheless, this approach was designed for network heterogeneity and was susceptible to single points failure to succeed. The (PCE) [20] was developed to allow customers to perform Router path computations, but it lacked a a specialized centralized path computation engine, as well as did not facilitate collaboration across many entities.(IRSCP) [21] introduced a An external router's path allocation module, enhancing dynamic interconnection for a single ISP service. Meanwhile, the 4D project [22] aimed to provide a clean-slate solution for introducing a topology discovery and traffic

forwarding logic control plane, but practical implementation was lacking. The SANE project [23], produced by the (NSF), aimed to enable traffic forwarding and access control regulations through the use of a within enterprise networks, a server that is logically centralized. Ethane, which succeeded the SANE project, offered an enhanced and improved control management module, with worldwide network awareness and pre-defined flow-based routing operations. However, both SANE and Ethane did not fully consider network components as a holistic entity and lacked flow-level regulation compared to conventional routing strategies. The control planes of these previous proposals were limited in terms of matching header fields and functionality. Consequently, SDN gained prominence with the advent of OpenFlow [16], which serves as a data-plane (API), and the introduction of a dependable centralized controller called NOX [6]. OpenFlow differs from earlier since it is an open protocol that empowers software designers to create applications on various switches supporting flow tables with a flexible set of header fields. SDN improves agility and flexibility by enabling server fast responsiveness to network changes, virtualization, policy deployment and centralized management of the entire network.

CLASSIFICATION AND COMPARISON OF SDN CONTROLLERS

To conduct a thorough comparison of various Controllers for SDN, we conducted an extensive review encompassing not only academic literature, as well as commercial sector. In this section, we initially outline the potential classification criteria for these controllers. Subsequently, we provide a comparative analysis and explore specific enhancements tailored to various use cases.

A. Classification & Selection Criteria

The operation of controllers is fairly consistent across Table I contains a collection of proposals. After evaluating 14 controllers, it becomes evident that the functioning, roles, as well as obligations of the the vast majority of these controllers don't offer a basis for classification. It appears that the only feasible classification criterion is the deployment strategy. Initially, the SDN concept aimed to consolidate the control plane, leading to the use of a single controller. However, this approach posed challenges in terms of scalability and a single point of failure. The architecture is dispersed. addresses these issues by enabling the implementation of multiple within a domain, domain controllers, organized in either a horizontal or hierarchical structure. It's important to note that in this study, we did'nt limit controller selection based on any special requirements. Instead, we gathered all possible controllers from the literature and documented projects. To the best of our ability, there is no existing work which has compiled and compared such a substantial total number of controllers.

B. Qualitative Comparison

Table I provides a comprehensive overview of various attributes associated with the controllers. Due to space limitations and the fact that not all proposals offer in-depth insights into their internal mechanisms, we refrain from discussing each controller on is own. Instead, we show you the characteristics as well as design decisions of these controllers.

Programming Language: Controllers have been developed using a variety a variety of programming languages, including C, C++, Java, JavaScript, Python, Ruby, Haskell, Go, and Erlang. some instances, a single language is used to build the entire controller. However, in numerous other controllers, There are several languages. employed within their foundation and modules. This approach is adopted to enable efficient memory allocation, compatibility with various platforms, and, the most significant, to achieve superior performance in the face of specific conditions.

Architecture: A critical design choice for a controller revolves around its structure, which may either be central or decentralized. Controllers that are centralized are primarily employed in smaller networks, in contrast to distributed controllers have the capability to extend spanning numerous domains. These distributed controllers can be categorized as either flatwhere all instances of controller share responsibility sharing, or even hierarchical, which includes a root administrator.

Programmable Interface (API): In broad terms, the (NBI) empowers the controller to support applications such as topological tracking, forwarding of flow, Virtualization of networks, load balancing, and detection of intrusions. These applications are driven by network events produced by data plane devices. Conversely, APIs at the lowest level, such as the Southbound API (SBI) have a distinct role.

TABLE I: SDN Controller's Feature Comparison Table

Name	Programming Language	Architecture	Northbound API	Southbound API	Interface	License	Multithreading	Documentation
Beacon [24]	Java	Centralized	ad-hoc	OpenFlow 1.0	CLI, Web UI	GPL 2.0	Yes	Fair
Floodlight [8]	Java	Centralized	REST, Java RPC, Quantum	OpenFlow 1.0, 1.3	CLI, Web UI	Apache 2.0	Yes	Good
FlowVisor [25]	C	Centralized	JSON RPC	OpenFlow 1.0, 1.3	CLI	Proprietary	-	Fair
HyperFlow [26]	C++	Flatly distributed	-	OpenFlow 1.0	-	Proprietary	Yes	Limited
NodeFlow [27]	JavaScript	Centralized	JSON	OpenFlow 1.0	CLI	Cisco	-	Limited
NOX [6]	C++	Centralized	ad-hoc	OpenFlow 1.0	CLI, Web UI	GPL 3.0	Yes (Nox-MT)	Limited
ONOS [10]	Java	Flatly distributed	REST, Neutron	OpenFlow 1.0, 1.3	CLI, Web UI	Apache 2.0	Yes	Good
OpenContrail [28]	C, C++, Python	Centralized	REST	BGP, XMPP	CLI, Web UI	Apache 2.0	Yes	Good
OpenDaylight[9]	Java	Flatly distributed	REST, RESTCONF, XMPP, NETCONF	OpenFlow 1.0, 1.3	CLI, Web UI	EPL 1.0	Yes	Good
OpenMul [29]	C	Centralized	REST	OpenFlow 1.0, 1.3, OVSDB, Netconf	CLI	GPL 2.0	Yes	Good
POX [7]	Python	Centralized	ad-hoc	OpenFlow 1.0	CLI, GUI	Apache 2.0	No	Limited
RunOS [30]	C++	Flatly distributed	REST	OpenFlow 1.3	CLI, Web UI	Apache2.0	Yes	Fair
Ryu [11]	Python	Centralized	REST	OpenFlow 1.0-1.5	CLI	Apache 2.0	Yes	Good
Yanc [31]	C, C++	Flatly distributed	REST	OpenFlow 1.0-1.3	CLI	Proprietary	-	Limited

Threading : A A single-threaded controller is more efficient appropriate for SDN that is lightweight implementations, whereas Controllers with multiple threads are available better suited for commercial applications such as 5G, SDN-WAN, and networks of optical fibers.

Licensing, accessibility, and documentation: The controllers covered in this article are as follows predominantly open-source, though a few are proprietary, restricting access to special requests or research purposes. Some controllers may not receive frequent updates due to difficulties in routine maintenance. Nonetheless, their source code is accessible online, enabling users to create necessary modifications. Our online investigation revealed that many controllers lacked adequate documentation. On the other hand, frequently updated controllers offer comprehensive and up-to-date documentation for all versions, along with community-based support.

Routing in SDN Environment

Achieving optimal routing is a critical objective in computer networks, and there are multiple methods to enhance routing based on various parameters. Moreover, the optimization of routing can differ depending on whether it pertains to regulate or transmit data. The centrally located nature of SDN controllers offers significant advantages compared to traditional routing methods. For instance, it allows for the easy extraction of network topology graphs and the dynamic calculation of efficient shortest-path algorithms like Dijkstra to identify the best routes. This Computer science algorithms are directly applied to computer networks[32] eliminates the requirement for They are being converted into distributed protocols and simplifies the creation of separate paths used in traffic engineering.

SDN also provides the flexibility to customize routing based on various criteria, including optimal routing types (e.g., shortest route, shortest constrained route), price functions, and resource considerations. This adaptability enables the seamless configuration and deployment of routing strategies tailored to specific scenarios, recognizing that a single routing approach may not uniformly suit all types of networks [33].

Related Works

The authors address the growing interest in Software Defined Networking (SDN) and the numerous OpenFlow controllers available for both research and commercial use. However, there is a lack of publicly available information regarding the architectural decisions that distinguish one controller's performance in real-world applications.

The primary aim of this research is to pinpoint critical performance limitations and highlight effective architectural decisions for creating efficient SDN controllers based on OpenFlow. To accomplish this objective, the authors evaluate the performance Includes four well-known open-source Controllers for OpenFlow: NOX, Beacon, Maestro, and Floodlight.To gauge the controllers' performance, the authors put them to use multi-threaded shared memory computers and subject them to benchmarking using various metrics, incorporating thread scalability, Scalability of switches, and delay. This assessment yields valuable insights and architectural guidelines that can enhance the current controllers' scalability guide the development of new ones. Drawing from these guidelines, the authors develop a new OpenFlow controller that surpasses existing controllers across different scalability metrics. Through the examination of architectural choices and performance evaluations, this research contributes to the enhancement of efficiency and scalability in OpenFlow-based SDN controllers. It provides valuable direction for researchers and developers in the SDN field, aiding them in the design and optimization of controllers for practical applications.

Khondoker , et al. (2014) delve into the challenge of selecting the most suitable SDN controller, such as POX, FloodLight, or OpenDaylight, for specific research needs. To assist researchers in making well-informed decisions, their paper introduces a decision-making template. The decision-making template operates as follows: Firstly, it conducts an analysis of existing open-source controllers to compile their properties. Subsequently, it employs a matching mechanism to align the properties of these controllers with the particular requirements of the researcher. For instance, if the researcher's requirement specifies a "Java" interface, the chosen controller must meet this criterion. Additionally, the study considers optional requirements, such as a preference for a GUI or the age of the controller. To evaluate controllers based on these optional criteria, the paper utilizes the (AHP), a technique for (MCDM). The AHP is modified by incorporating a Mechanism of monotonic interpolation/extrapolation that relates the values of controller attributes to a predefined scale. Using this adapted AHP, the research compares the top five controllers and concludes that "Ryu" emerges as depending on the best suited controller on their exact specifications [34].

Salman , et al. (2016) recognize the development of numerous controllers and the various studies conducted to assess and compare their performance in recent years. Importantly, their paper introduces new controllers, specifically ONOS Controllers based on libfluid (raw, base), and evaluates their performance Cbench is being used, an OpenFlow testing software. While the outcomes highlight that MUL and Beacon controllers demonstrate the highest performance, the research underscores the importance of selecting the most suitable controller based on a range of criteria that align with user requirements. In essence, different controllers may excel in distinct aspects, underscoring the significance of researchers or users choosing the controller that best fits their specific needs and objectives [35].

Zhu , et al. (2019) underscore the importance of accurately assessing and benchmarking the performance and capabilities of SDN controllers. Despite numerous controller proposals in existing literature, there's a noticeable absence of quantitative comparative analysis. To bridge this gap, the research delivers an extensive qualitative contrast of various SDN controllers and conducts a measurable examination in terms of their performance across diverse situations for networks. They Classify and categorize 34 controllers determined by their functionalities and provide a high-quality assessment of their characteristics. Furthermore, the study delves into the benchmarking tools' capabilities designed for evaluating SDN controllers and outlines effective methods for quantitatively assessing these controllers. The research employs three benchmarking instruments evaluate there are nine controllers based on a number of criteria. Overall, this research provides comprehensive insights into the performance, benchmarking standards, and assessment setups for SDN controllers. By shedding light on the capabilities and performance of different controllers, this study makes a valuable contribution to the field of SDN and offers guidance to network practitioners and researchers [36].

SHARIF, et al. (2020) highlight the critical importance of precise and effective benchmarking tools for assessing controller performance across different evaluation parameters. Despite the presence of numerous controller proposals in both general and specialized network domains within the literature, there exists a noticeable lack of comprehensive quantitative analysis for these controllers. To address this gap, the research offers an extensive qualitative contrast of various SDN controllers and conducts quantitative evaluations of their efficiency under diverse situations for networks. They Classify and categorize 34 and provide a qualitative comparison of their attributes. Furthermore, the study includes a controller comparison analysis specifically designed in the case of specialized networks, including the (IoT), Networks based on blockchain, transportation networks, and Sensor networks that operate wirelessly. This comprehensive approach provides a deeper understanding of controllers tailored for specific use cases. Additionally, the authors delve into the benchmarking tool capabilities and conduct a detailed comparison analysis to select the most suitable tools for performance assessment. The research employs three benchmarking tools to evaluate nine controllers, resulting in a comprehensive performance analysis for each controller and offering insights into the performance of specialized network controllers. By delivering a comprehensive analysis of SDN controllers and their performance across various network scenarios, this research contributes valuable insights to the field of SDN. It provides a holistic understanding of controller capabilities and benchmarking methodologies, offering valuable guidance for both researchers and practitioners in the field of SDN [37].

Gupta, et al. (2022) The authors discuss the development of various SDN controllers, such as Beacon, Floodlight, Ryu, ODL, ONOS, NOX, and Pox, to cater to the diverse range of SDN applications and requirements. The research underscores the importance of selecting the most appropriate controller based on specific application needs. The paper explores the evolution of networking

architecture from a fully scattered form to a more concentrated one with the advent of SDN. It assesses and compares the effects many types SDN controllers on SDN. The study places particular emphasis on SDN controllers, considering them as the "brains" of the network, and delves into their distinctions to identify the most optimal controller overall. By using the simulation environment Mininet the research conducts a performance comparison of SDN controllers like Ryu, ODL, and others. It presents the experimental findings, demonstrating how ODL in comparison to other controllers various network architectures, including both standard and customized topologies integrated with ODL. ODL is recognized as the preferred controller due to its superior bandwidth and reduced latency [38].

Discussion

In this study, we explored Software Defined Networking controllers and their significance in the SDN architecture. SDN controllers serve as the central element responsible for managing and orchestrating the data plane, making their performance and capabilities crucial for optimizing network operations. Our objective was to determine the most appropriate SDN controller for our particular network scenario, and after a comprehensive evaluation, we chose the Ryu controller for several compelling reasons. When selecting an SDN controller, several key criteria come into play, including performance, scalability, flexibility, ease of use, and community support. The chosen controller should align with the specific requirements and objectives of the network deployment. The process of evaluating and comparing SDN controllers is essential to make an informed decision and ensure that the selected controller meets the needs of the network environment. Our evaluation involved a comparative analysis of several prominent SDN controllers available in the literature, including Beacon, Floodlight, Ryu, ODL, and others. Each controller was assessed based on its strengths and weaknesses, and how well it aligns with our specific evaluation criteria. After careful consideration and benchmarking, the Ryu controller emerged as the most suitable option for our network scenario. Ryu demonstrated exceptional performance in terms of scalability, flexibility, and compatibility with our network environment. It provided robust support for various protocols, making it highly adaptable to our network's evolving needs. Furthermore, Ryu offered extensive community support and active development, ensuring continuous updates and improvements.

Conclusion

We conducted a comprehensive assessment of 14 publicly accessible OpenFlow controllers, thoroughly examining their architectural designs to identify their respective strengths and weaknesses. Our findings provided valuable insights and guidelines that can be utilized to achieve consistent enhancements in the performance of SDN controllers. Additionally, we developed a new controller based on the derived guidelines, which demonstrated significant improvements in terms of throughput, latency, and switch scalability. The proposed controller's performance consistently outperformed the existing ones, showcasing its potential for more efficient and effective network operations.

REFERENCES

[1] C. Y. Hong *et al.*, “B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google’s software-defined WAN,” in *SIGCOMM 2018 - Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, Association for Computing Machinery, Inc, Aug. 2018, pp. 74–87. doi: 10.1145/3230543.3230545.

[2] S. Bera, S. Misra, and A. V. Vasilakos, “Software-Defined Networking for Internet of Things: A Survey,” *IEEE Internet Things J*, vol. 4, no. 6, pp. 1994–2008, Dec. 2017, doi: 10.1109/JIOT.2017.2746186.

[3] V. G. Nguyen, A. Brunstrom, K. J. Grinnemo, and J. Taheri, “SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1567–1602, Jul. 2017, doi: 10.1109/COMST.2017.2690823.

[4] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, “Privacy-Preserving DDoS Attack Detection Using Cross-Domain Traffic in Software Defined Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628–643, Mar. 2018, doi: 10.1109/JSAC.2018.2815442.

[5] R. Mungara#, K. Venkateswararao#, and V. Pallamreddy#, “A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks.” [Online]. Available: www.ijcta.com

[6] N. Gude *et al.*, “NOX: Towards an Operating System for Networks.” [Online]. Available: http://www.noxrepo.org

[7] “POX Controller Manual Current Documentation.” [Online]. Available: https://noxrepo.github.io/pox-doc/html/”.

[8] “Big Switch Networks, “Project Floodlight.” [Online]. Available: http://www.projectfloodlight.org/floodlight/”.

[9] ““OpenDaylight: A Linux Foundation Collaborative Project.” [Online]. Available: https://www.opendaylight.org/”.

[10] P. Berde *et al.*, “ONOS: Towards an open, distributed SDN OS,” in *HotSDN 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking*, Association for Computing Machinery, 2014, pp. 1–6. doi: 10.1145/2620728.2620744.

[11] “Ryu SDN Framework Community, “Ryu Controller.” [Online]. Available: https://osrg.github.io/ryu/index.html”.

[12] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4. Institute of Electrical and Electronics Engineers Inc., pp. 2181–2206, Apr. 24, 2014. doi: 10.1109/COMST.2014.2326417.
[13] K. Dhamecha and B. Trivedi, "SDN Issues-A Survey," 2013.
[14] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.0440
[15] "A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for highlevel reactive network control," in Proc. 1st Workshop Hot Topics Softw. Defined Netw., Helsinki, Finland, 2012, pp. 43–48.".
[16] N. Mckeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," 2008.
[17] "The SoftRouter Architecture."
[18] "Forwarding and Control Element Separation (ForCES) Framework".
[19] "N. Feamster, H. Balakrishnan, J. Rexford et al., "The Case for Separating Routing from Routers," in Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, 2004, pp. 5–12.".
[20] "A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC 4655, Internet Engineering Task Force,2006.".
[21] J. Van Der Merwe *et al.*, "Dynamic Connectivity Management with an Intelligent Route Service Control Point," 2006.
[22] A. Greenberg *et al.*, "A Clean Slate 4D Approach to Network Control and Management *," 2005.
[23] O. Akonjang, "SANE: A Protection Architecture For Enterprise Networks," 2006.
[24] "D. Erickson, "The beacon openflow controller," Proceedings of the second ACM SIGCOMM workshop, pp. 13–18, 2013.".
[25] R. Sherwood *et al.*, "FlowVisor: A Network Virtualization Layer," 2009. [Online]. Available: http://OpenFlowSwitch.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf
[26] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow."
[27] "NODEFLOW: An openflow controller node style." [Online]. Available: http://garyberger.net/?p=537".
[28] "OpenContrail An open-source network virtualization platform for the cloud." [Online]. Available: http://www.opencontrail.org/.
[29] "OpenContrail An open-source network virtualization platform for the cloud." [Online]. Available: http://www.opencontrail.org/
[30] "GitHub, "RunOS OpenFlow Controller." [Online]. Available: https://github.com/ARCCN/runos".
[31] M. Monaco, O. Michel, and E. Keller, "Applying Operating System Principles to SDN Controller Design," Oct. 2015, doi: 10.1145/2535771.2535789.
[32] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 1, pp. 388–418, Jan. 2018, doi: 10.1109/COMST.2017.2749760.
[33] D. Lopez-Pajares, E. Rojas, J. A. Carral, I. Martinez-Yelmo, and J. Alvarez-Horcajo, "The Disjoint Multipath Challenge: Multiple Disjoint Paths Guaranteeing Scalability," *IEEE Access*, vol. 9, pp. 74422–74436, 2021, doi: 10.1109/ACCESS.2021.3080931.
[34] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers."
[35] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," in *Proceedings of the 18th Mediterranean Electrotechnical Conference: Intelligent and Efficient Technologies and Services for the Citizen, MELECON 2016*, Institute of Electrical and Electronics Engineers Inc., Jun. 2016. doi: 10.1109/MELCON.2016.7495430.
[36] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN Controllers: Benchmarking & Performance Evaluation," Feb. 2019, [Online]. Available: http://arxiv.org/abs/1902.04491
[37] L. Zhu *et al.*, "SDN Controllers: A Comprehensive Analysis and Performance Evaluation Study," *ACM Computing Surveys*, vol. 53, no. 6. Association for Computing Machinery, Feb. 01, 2021. doi: 10.1145/3421764.
[38] N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen, and S. Bharany, "A Comparative Study of Software Defined Networking Controllers Using Mininet," *Electronics (Switzerland)*, vol. 11, no. 17. MDPI, Sep. 01, 2022. doi: 10.3390/electronics11172715.