Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.047009

REVIEW

Open-Source Software Defined Networking Controllers: State-of-the-Art, Challenges and Solutions for Future Network Providers

Johari Abdul Rahim[1], Rosdiadee Nordin[2,*] and Oluwatosin Ahmed Amodu[3,4]

[1]Technology Strategy and Fundamental Planning (TSFP), Telekom Malaysia, Cyberjaya, Selangor, 60000, Malaysia

[2]Department of Engineering, School of Engineering and Technology, Sunway University, 5, Jalan Universiti, Bandar Sunway, Selangor, 47500, Malaysia

[3]Department of Electrical, Electronics & Systems Engineering, Universiti Kebangsaan Malaysia, UKM Bangi, Selangor, 43600, Malaysia

[4]Information and Communication Engineering Department, Elizade University, P.M.B. 002, Ilara-Mokin, 340271, Nigeria

*Corresponding Author: Rosdiadee Nordin. Email: rosdiadeen@sunway.edu.my

Received: 21 October 2023 Accepted: 08 March 2024 Published: 18 July 2024

ABSTRACT

Software Defined Networking (SDN) is programmable by separation of forwarding control through the centralization of the controller. The controller plays the role of the 'brain' that dictates the intelligent part of SDN technology. Various versions of SDN controllers exist as a response to the diverse demands and functions expected of them. There are several SDN controllers available in the open market besides a large number of commercial controllers; some are developed to meet carrier-grade service levels and one of the recent trends in open-source SDN controllers is the Open Network Operating System (ONOS). This paper presents a comparative study between open source SDN controllers, which are known as Network Controller Platform (NOX), Python-based Network Controller (POX), component-based SDN framework (Ryu), Java-based OpenFlow controller (Floodlight), OpenDayLight (ODL) and ONOS. The discussion is further extended into ONOS architecture, as well as, the evolution of ONOS controllers. This article will review use cases based on ONOS controllers in several application deployments. Moreover, the opportunities and challenges of open source SDN controllers will be discussed, exploring carrier-grade ONOS for future real-world deployments, ONOS unique features and identifying the suitable choice of SDN controller for service providers. In addition, we attempt to provide answers to several critical questions relating to the implications of the open-source nature of SDN controllers regarding vendor lock-in, interoperability, and standards compliance, Similarly, real-world use cases of organizations using open-source SDN are highlighted and how the open-source community contributes to the development of SDN controllers. Furthermore, challenges faced by open-source projects, and considerations when choosing an open-source SDN controller are underscored. Then the role of Artificial Intelligence (AI) and Machine Learning (ML) in the evolution of open-source SDN controllers in light of recent research is indicated. In addition, the challenges and limitations associated with deploying open-source SDN controllers in production networks, how can they be mitigated, and finally how open-source SDN controllers handle network security and ensure that network configurations and policies are robust and resilient are presented. Potential opportunities and challenges for future Open SDN deployment are outlined to conclude the article.

KEYWORDS

ONOS; open source software; SDN; software defined networking

1 Introduction

The technology industry is experiencing a major revolution by moving towards a process that switches from closed (or proprietary) to open source-based products and services. An open-source survey done by BlackDuck in 2016 shows that more than 65% of companies adopt open-source software and 67% see the value in open source by actively engaging in open-source projects [1,2]. Previous research works like [3–8] have shown that the level of adoption growth of open-source projects has increased exponentially. The adoption growth is continuously increasing year by year. Without a doubt, free and open-source-based technology has become the core of most technologies today. Major software giants use and promote open-source technology including Google, Twitter, Facebook, and other technology companies [1,2]. Some of the giant technology companies, such as Nippon Telegraph and Telephone (NTT), Big Switch Networks, Nicira Networks, Red Hat, and Cloudera have gone public in advocating the industrial focus on open source-based technology [1,2]. These companies offer products and solutions based on open-source software and most of these companies play a big part in developing and maintaining the open-source projects.

With the support and involvement of technology partners, new product innovations based on open-source projects have significantly increased in respective communities. Commonly, in Information and Communications Technology (ICT) industries, there is a trend of adopting open-source solutions in the telecommunication and networking industry. Global Developer Survey 2016 by GitLab revealed that open-source solution is the most selected tool for developers and companies are using the solution as the core underlying technology [3]. Currently, open-source solutions are part of the largest world network solutions. Therefore, Software Defined Networking (SDN) solutions have become one of the fastest-growing solutions. Many software solutions practically applied in SDN are offered by open-source controllers through open standards and platform neutrality. For the past few years, tremendous growth has been observed in SDN-based open-source projects, including Open Network Operating System (ONOS). In addition to ONOS released to cater for the SDN industry, there are many other controllers such as the first OpenFlow Network Controller Platform (NOX), Python-based Network Controller (POX), Floodlight, and OpenDaylight (ODL). ONOS is an open-source solution designed specifically for the carrier-grade network while offering intelligence into the software controller through open standard interfaces. Supported by dedicated developers and technology leaders around the world, the ONOS project offers a distributed architecture of controllers to accommodate the reliability and scalability of service providers' networks. By centralizing management network resources and a high-level abstraction layer with a standard Application Programming Interface (API), the platform provides the ability to create network applications and services over open hardware while provisioning resources through an open interface.

Although SDN is a relatively new concept in network architecture, it is being heralded as a fast solution to all network infrastructure problems (see Fig. 1). The strengths of SDN are characterized by the separation of control and data plane in the network, and enabling the network intelligence into a centralized controller. The separation of software network logic and its infrastructure is possible with SDN programmable control and management. The concept of network intelligence of controllers is logically centralized and the abstraction of applications away from the network infrastructure. The network devices are separated from the applications by the controller between them. All communications between applications and devices must go through the controller. Similar to an operating system, the SDN controller functions by facilitating automated network management, empowering network administrators on network functionality, and presenting a unified view of the network. In other words, SDN is not only one specific solution, technology, or product but a range of advances in the next generation of networking. From the solid growth of open source solutions,

there are quite many SDN controllers available in the open market with the latest innovation that is reliable and service provider-focused. Open-source evolution enriches the SDN ecosystem and plays a significant role in the openness of SDN controllers. There is a variety of open-source SDN controllers, from POX to the latest popular ONOS, an open-source community hosted by the Linux Foundation. Given the above, this paper provides a thorough review of the open-source solutions for SDN controllers.

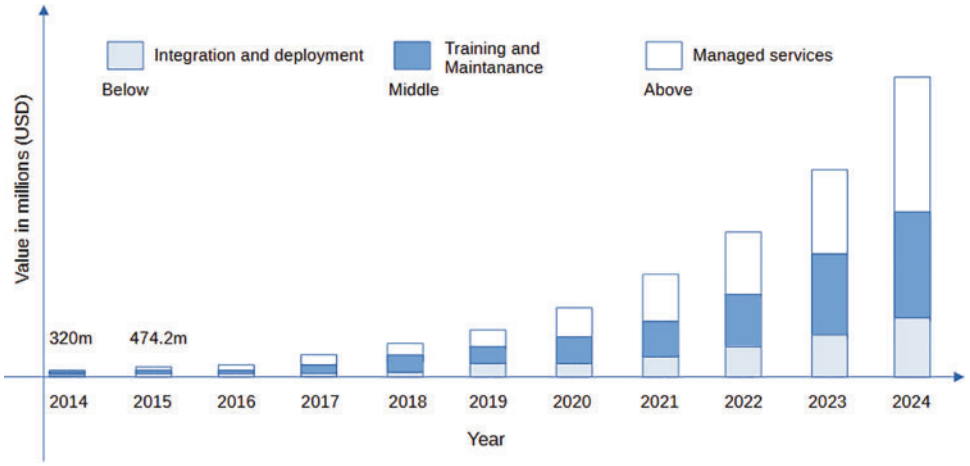

Figure 1: Global software-defined networking market by services, 2014–2024, based on [9]

1.1 Related Surveys

Some reviews have appeared in the literature related to the subject of open SDN controllers. For in-stance, the authors in [10] briefly reviewed the architectural alternatives for open-flow and open-source SDN controllers which include Beacon, Nox, Maestro, and Floodlight which support multi-thread concepts. They briefly summarize the issues at different SDN layers such as storage, processing, hardware platform, performance evaluation, programming language, network measurement and synchronization, adaptive routing, ease of maintenance, security, etc. Then they briefly discuss different SDN controllers (and their evolution) such as OpenDayLight, Ryu, Trema, OMNI, McNettle, and SNAC while providing the application features of these controllers. They discuss the architecture of the OpenFlow switch testbed and different SDN projects and tools, metrics, and their benefits concerning the evolution of SDN controllers.

The authors in [11] provide a comprehensive survey where they discussed the difference between traditional network architecture and SDN-based architecture as well as the best feasible SDN implementations and favorable SDN controller selection. In addition, criteria for performance assessment of SDN OpenFlow controllers were discussed. Furthermore, research works on OpenFlow-based controllers, issues regarding multifarious network conditions, metrics, scalability to network load, and various topology designs were outlined. The theme of this work was to outline the research gaps concerning the performance limitations of the controller while providing suggestions for an optimized solution.

The intelligent control of how incoming packets are forwarded through forwarding devices when they arrive in the network is carried out by the centralized SDN control plane which also provides a bird's-eye view of the entire network via a single central point. Three deployment models can be considered to implement the control plane. A single-controller network configuration (physically

centralized), a multi-controller configuration for network management (logically centralized but physically distributed), and a hybrid co-existence of both traditional centralized and distributed control. Ahmad et al. [12] studied these different control plane architectures while discussing SDN controllers that support the discussed architectures. The authors analyze over forty SDN controllers in terms of diverse performance measures such as how they scale (scalability), how reliable they are (reliability), how consistent they are (consistency), and how secure they are (security). The mechanisms for achieving these measures, as well as their merits and demerits, were highlighted. In addition, the authors identified challenges and future research opportunities for diverse SDN control plane architectures.

The authors in [13] presented a survey where they compared traditional networking and SDN-based networking. The paper spans SDN controllers, how they are deployed in the current internet paradigm, the OpenFlow architecture and configuration, SDN security (including threats and solutions) rules, illegal access, attacks on IoT devices, and hardware trojan attacks. Also, the paper provides details about different SDN approaches and how they differ or are similar to each other. Then the authors discussed SDN applications (such as their use in rural environments, mobile device offloading, upgrading of data centers, network virtualization, etc.) and benefits to provide readers with a wider range of prospects while providing a summary of how SDN emulation tools and testbeds.

In [14], the authors conducted a review of the early works on network programmability into the telephone network and its progress over several decades, particularly including rapid advances in the later part of the 1990s while describing the most noteworthy works. The importance of the architectural evolution and its influence on modern and next-generation computing was also detailed, providing a critical perspective on how industry and academia have collectively failed to institute network programmability long before. The arrival of the OpenFlow standard and the significance of the architectural transformations are also provided.

1.2 Objectives and Contribution

In contrast to the aforementioned works, this paper presents two main contributions.

Firstly, as a background, a comparative and comprehensive study on open-source SDN controllers such as NOX, POX, Ryu, Floodlight, ODL, and ONOS. It also reviews the architecture and evolution of the ONOS controller. The review includes the deployed application and use cases based on the ONOS controller. The advantages and limitations of open source SDN controllers are discussed, exploring carrier-grade ONOS for future real-world deployments, ONOS unique features, and the differences with other SDN controllers while identifying suitable choices of SDN controllers for service providers. The study will also, identify significant parameters that can contribute to determining the suitable Open SDN controller for their specific application and services. Also, it will identify the characteristics and evolution of the carrier-grade ONOS controller that is built specifically for the service provider environment.

Secondly, the paper discusses recent advances in the performance evaluation, monitoring, topology, architecture, and security of open SDN controllers. In addition, the implications of the open-source nature of SDN controllers, vendor lock-in, interoperability, and standards compliance; real-world use cases and success stories of organizations implementing open-source SDN controllers in their networks; how the open-source community contributes to the development and improvement of SDN controllers are discussed as well as the key challenges faced by open-source SDN projects. Similarly, considerations organizations should take into account when choosing an open-source SDN controller for their specific networking re- requirements and goals are provided, and how emerging

technologies like Artificial intelligence (AI) and Machine Learning (ML) impact the evolution and capabilities of open-source SDN controllers. In addition, the challenges and limitations associated with deploying open-source SDN controllers in production networks, as well as how they can be mitigated, and finally, how open-source SDN controllers handle network security while ensuring network configurations and policies are robust and resilient, are discussed.

Potential opportunities and challenges for future Open SDN deployment have also been highlighted. The organization of this paper is based on the structure in Fig. 2.

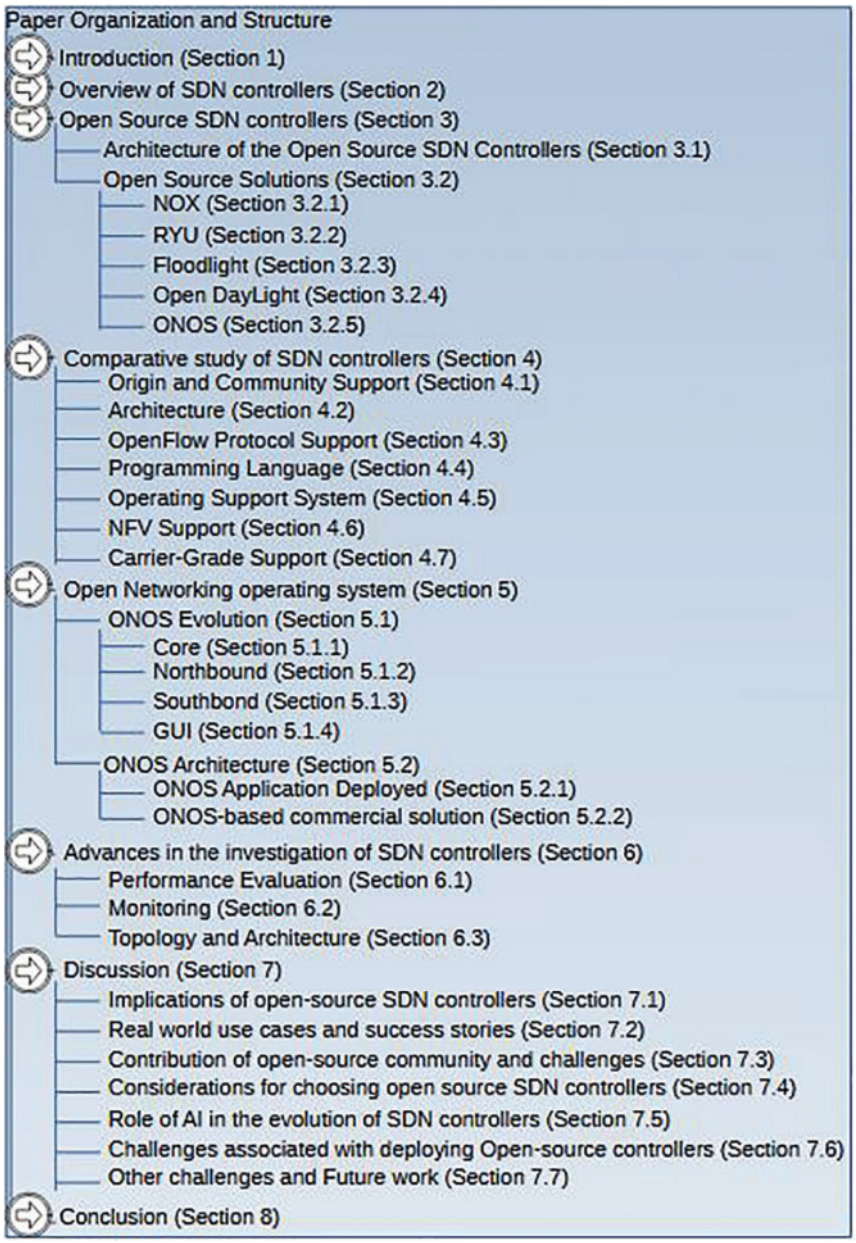

Figure 2: Organization of the paper

2 Overview of the SDN Controller

The SDN controller plays a central role in the network as it enables programmable networking by abstracting the control plane from the physical network devices. SDN controller consists of a series of modules and plugins that can perform various network tasks and collect network statistics by abstraction network layers. To market the solution while enhancing the capabilities and features of the SDN controller, developers build their version of the controller with unique network algorithms and preferred standards or protocols. The SDN controllers can either be proprietary or open-source SDN controllers.

- SDN controller vendors: vendors build exclusive software and hardware solutions that are based on commercial license and profit. Vendors-based SDN controller may be built upon open source platform but the end product is highly fine-tuned by the vendor. The vendor provides stable products and maintains high-quality control with tailored support. A few examples of vendor-based SDN controller products are BluePlanet, Cisco WAE Juniper NorthStar, and Contrail.
- Open source SDN controllers: The open source SDN controllers are developed and maintained by a distributed community of individual parties, organizations, non-profit foundations, or developers who work together without being cost-oriented. Some of these projects are inherited directly from proprietary solutions and others start as distributors of license-free packages. A few examples of community open-source SDN controller projects are Ryu, Opendaylight, and ONOS.

SDN controller solutions are grouped into two categories: proprietary and open-source-based. Open source controllers have shown incredible progress in the SDN domain and present great functionalities and potentials, including network virtualization and network function virtualization. Open SDN controllers, however, have their constraints such as:

- No Warranty: The software license is open source which means that there is no warranty provided. On the other hand, warranty protection is customarily provided for commercial products.
- Intellectual Property Infringement: Since it is developed without the usual commercial controls, theoretically, any developer can add infringing code.
- Incompatible or Uncompleted Projects: No motivation and clear objective that could drive project delivery and provide continuous solutions.
- Security Issues: Open source code can be read and compromised in principle. It may not be following security guidelines and lacks data encryption.
- Lack of Professional Technical Support: Business-level support expects 24/7 response time and most of the open source solutions do not come with this unless the company subscribes to a premium support package from the project leading company or a separate third party.
- Lack of Direction in Product Development: Some of the projects do not have a clear direction and only focus on resolving the specific problem statement, which is not relevant to the latest environment in the long run. Some of them are designed with limited features for training and research purposes. NOX and Beacon, for example, are SDN controllers that are designed for research purposes and are no longer maintained or updated [15].
- Higher Indirect Cost: There is a high indirect cost such as deployment, integration, and maintenance with different vendors
- Not Entirely Free: Some of the features (limited or full capabilities) require a paid version.
- High Learning Curve: The learning curve is higher as the solution provides the required multi-skills and good technical knowledge. Proof of concept and simulation can be performed without

limitation as no license is required. For example, a list of ONOS released can be downloaded from the public repository at any time without any restriction. A specific release can be tested based on the features required.

- Lack of Proper Documentation and References: Open source packages are released regularly with enhancement features or patches released for bug updates. Each component resource or a guideline of installation must be documented for ease of future reference. These activities require a dedicated team for management and control. Proper and updated documentation is also required in any open- source project.

Most SDN users are still thoughtful of open-source software-based networking solutions. Meanwhile, quite several SDN vendors are pushing for commercial SDN that is simpler to realize and manage. This is not always true. Commercial SDN is a closed-source arrangement that protects the products and is keen to maintain control over the solution and user experience. Most of the commercial solutions are distributed without the source code, and may not be fully interoperable with other network solutions and network devices.

Open source controllers enable a modular-based framework composed of plug-in modules that can dy-dynamically carry out various tasks and promote network function virtualization and cloud computing readiness. Programming code is validated and tested hence increasing confidence in using the SDN controller open source solution. The numerous open-source SDN controllers available to the user are different from each other in several aspects. Open source, open standards, and open protocols are essential to SDN. Networking is basically defined as an activity that connects related systems. Thus, certain criteria must be outlined to administer this communication. In any network infrastructure, network traffic is managed and controlled by closed proprietary hardware that runs its network functions through proprietary software and some are partly operated based on a combination of proprietary software and open-source software. Through APIs and open standards, SDN restrictions no longer exist. As APIs for business applications and services become more common, other opportunities such as Network Function Virtualization (NFV) of the networking infrastructure have been made possible by SDN. SDN controllers are able to integrate cloud-based services and together catalyst more commodity hardware and more open software through network function virtualization capabilities. Most of the open-source SDNs listed below work well with NFV environments, for example in the data centre and cloud use cases [16–18].

Based on a report made at the 2016 Open Session Summit in Santa Clara, CA, open sources impact the telecommunication and network industry. Even at the previous meeting, the focus was more on the areas of SDN. Among the key indicators of increasing open source impact are the level of adoption and development of new open source-based products. Another is how companies that formerly utilized the resources now turn to contributors for the development of projects which eventually became the basis of a global initiative to the success of great technology companies.

There are many open-source planning and activities focused on building a controller SDN. These activities are supported by many parties and one of them is Open Networking Lab (ON.Lab), a Non-Profit Organization (NGO) founded by professionals from Stanford University and UC Berkeley. ON.Lab runs many research projects and one of them is ONOS. ONOS is built on open source specific for service providers. Another open-source activity focusing on building SDN controllers is the OpenDayLight Project (ODL). Hosted by the Linux Foundation, ODL, established in 2013, aims to provide open-source frameworks and an SDN platform set up with a solid foundation for NFV.

Both ONOS and ODL projects are hosted by the Linux Foundation, the non-profit organization dedicated to the development of ecosystems through open source to advance technology development

and in-increase usage rates in the industry. The great benefits to the success of SDN-based projects are the ability to join and collaborate with the open source community, leveraging on the entire community to build better products and bring them to market faster. For example, there is tremendous momentum behind SDN projects like ONOS. SDN acceptance can be achieved with a deep interest in SDN's open-source project. In line with that, in 2015, the Open Network Foundation (ONF) introduced OpenSourceSDN.org (accessed on 10/01/2024) to support and develop a more comprehensive open SDN solution. Like open-source organizations such as ONOS, ODL, ON.Lab and Open Platform for NFV (OPNFV), close collaboration with OpenSourceSDN.org in launching missions and objectively expanding SDN-related businesses. Many community organizations are trying to figure out what will ultimately be the basis for moving forward and leading to success. Open source has become the dominant model for how the world's SDN technology is built and operated. Although the above-mentioned comparison deems open-source controllers to be a more suitable choice than the commercial proprietary-based platform, providers are still divided regarding whether to deploy open-source solutions or proprietary-based platforms.

Many surveys and research have been performed to understand why the open-source solution is becoming popular and good choice [19,20]. The survey of Gigaom Research [21] reported that the majority of the 600 survey respondents will have deployed SDN solutions for the next 1–3 years. According to research firm HIS in 2016, more than 75% of operator respondents have deployed or will deploy SDN in 2016–2017 [22]. These aggressive timelines reflect high hopes for SDN solutions and also closely related open systems and open-source technologies. The findings also show that network operators use a variety of suppliers to avoid vendor lock-in. Even though commercial or proprietary SDN controller is still the most accepted solution, many deployments and proof of concept have demonstrated that open-source SDN solution has the capabilities and technologies and has matured into something even more powerful and useful. Table 1 compares open source and vendor-based/proprietary SDN controllers. In the next sections, open-source SDN controller platforms will be examined.

Table 1: Comparing vendor-based/proprietary and open-source SDN controller

	Open Source SDN controller	Vendor/Proprietary SDN controller
Vendor-lock in	A neutral choice with no favouritism of platform and vendor.	Strong attachment to vendor's technology and solution
	Free solution and very minimal cost for community fee membership if required.	Costly and each service comes with a rate per unit.
Cost	No cost of licenses, software upgrades and patches.	Upgrading, customization, and integration work would require specific costs and needs to renew high operation and maintenance costs.
	Minimum operating cost.	Additional services, modules, and upgrades will cost more.

(Continued)

Table 1 (continued)

	Open Source SDN controller	Vendor/Proprietary SDN controller
Limitation/ Restriction	No restriction.	Close architecture and enhancement /tuning/hardening that do not work well with different platforms or technologies.
	Ability to customize based on actual business needs Open architecture and interface concept.	Source codes are not shared. Limitations or restrictions also may be due to licensing that must be purchased.
Support	Public community.	Strong support from vendors based on what has been purchased or maintenance job scope.

3 Open Source SDN Controllers

3.1 Architecture of the Open Source SDN Controller

SDN architecture consists of three layers: application (application layer), control plane (controller layer), and data plane (network/data layer). The first one is an application layer where software programs reside to perform network behavior tasks. The second is the control plane in charge of the determination of an optimal path for data flow, and responsible for making network forwarding decisions. Routing or data forwarding does not involve the management plane. The network administrative information oversees network programmability and administration duties such as OpenFlow, NetConf, and BGP [23,24]. The third plane known as the data plane turns network devices into just simple forwarding devices while providing programmability instead of configuration mode. Network programmability designed and built by SDN refers to the potential to initialize, change, control, and manage network behavior dynamically via open interfaces. As shown in the figure below, SDN controllers are connected through northbound interfaces (NBI) and southbound interfaces (SBI). Fig. 3 displays the SDN in high-level architecture. This section will provide a general overview of the Open Source SDN Controllers.

3.2 Open Source Solutions

There are several SDN solutions have been developed for the adoption of open-source SDN controllers [25–27]. Among these, are NOX, POX [28,29], Ryu [30], Floodlight [31], OpenDaylight [32] and ONOS [33,34]. A comparative study is performed to explore the available open-source SDN controllers and understand why ONOS controllers could be the highest choice of service providers. Listed below are free and open-source SDN controllers. Here we will briefly discuss each of these controllers. Fig. 4 shows the five different categories of open-source SDN controllers, and Fig. 5 illustrates their APIs.

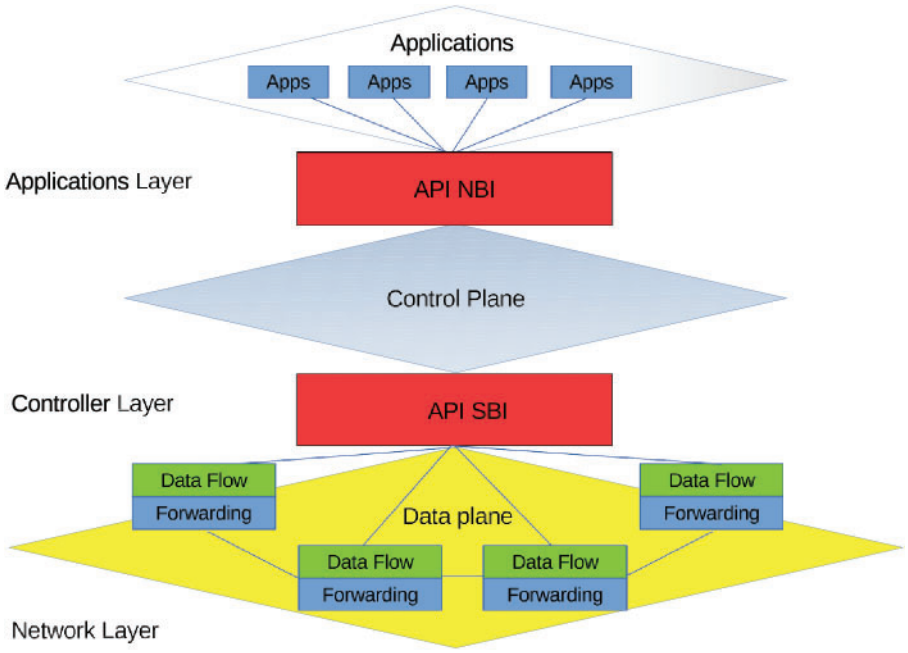

Figure 3: Layered model and architecture

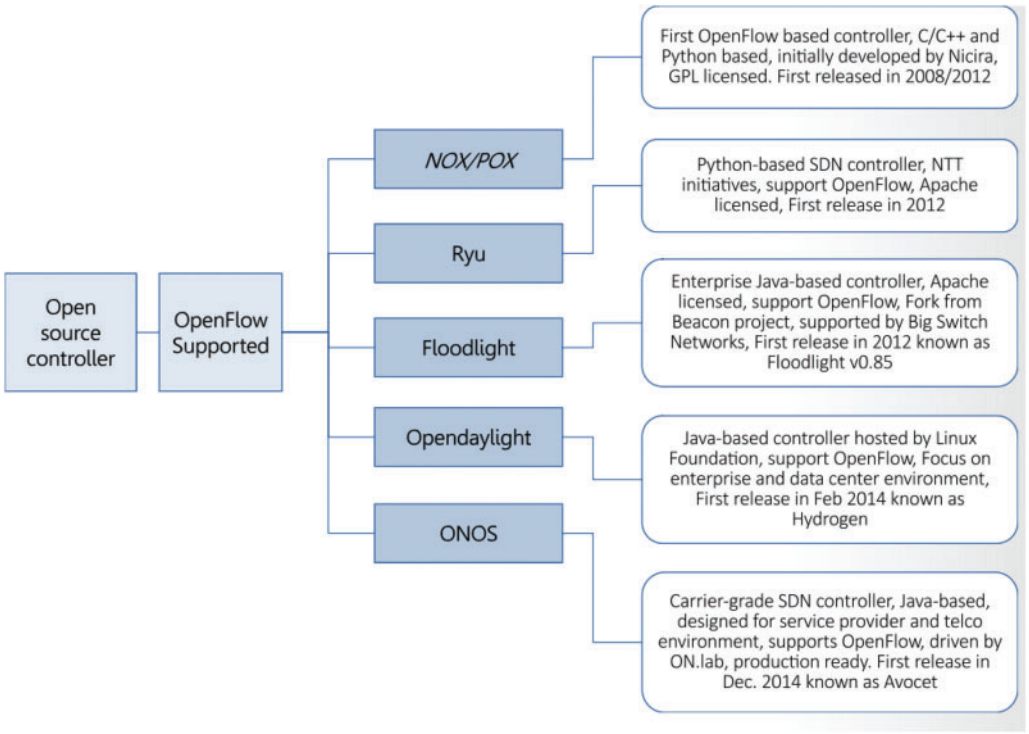

Figure 4: Open source controllers

Figure 5: SDN controller API

3.2.1 NOX

NOX was the first Open Flow-based SDN controller [28], initially developed by Nicira and later became open source in 2008. It then expanded and was supported by ON. Lab activity at Stanford University, UC Berkeley and ICSI. In 2011, the people responsible for constructing the frame of SDN collectively established the Open Network Research Center (ONRC) and ON.Lab to concentrate, improve, deploy, and support open-source SDN tools and platforms. NOX has two versions:

a) NOX Classic: The version that has been available since 2009 under GPL. It is a network control platform based on C++ and supports Python. However, its line of development has been suspended.
b) NOX: The "new NOX." This version only supports C++ with less network application but it has greater speed and provides a better codebase compared to NOX-Classic.

However, NOX has its limitations, for example, a new platform, POX had to be created to address these problems. Known as the younger sibling of NOX, the POX controller is a pure Python version of NOX [28].

3.2.2 Ryu

Ryu (which means dragon in Japanese) is built on a component-based SDN framework, Ryu, flows and a school of thought). Ryu is fully written in Python and supported by NTT labs [30]. Its packages with software components and well-defined API make it one of the popular SDN platforms used by the developers. It supports organizations to customize deployments to meet their specific needs. On the Southbound protocols, Ryu supports multiple protocols as plugins, e.g., OpenFlow 1.0 to 1.5, Netconf, OFconfig, etc. In addition, Ryu also supports Nicira Extensions. Maintains and manages by an open community and hosted on GitHub, Ryu source codes are freely available and usable. The deployment of Ryu as a network controller is also supported by OpenStack, an open-source cloud operating system. This support is positive for Ryu as OpenStack can control the computing, storing, and networking of a pool of hardware resources.

3.2.3 Floodlight

The Floodlight Controller is an Open SDN Controller. It was originally developed at Standford University and UC Berkeley [31]. It now continues to be supported by a community of open-source developers including experts from Big Switch Networks. Floodlight works with OpenFlow to manage data flow by allowing the SDN controller to instruct the forwarding plane to modify the behavior

of network devices. Floodlight is designed to work with routers and physical and virtual switches that make up the forwarding plane. It is also very flexible since it can work in a variety of environments, even supporting network groups that are a hybrid of OpenFlow compatible and non-OpenFlow switches. As it is written in Java, Floodlight adapts easily to software and develops applications. The controller may be freely used, reproduced, modified, distributed or sold as Floodlight is released under a free and open-source software (FOSS) licensing agreement from the Apache Software Foundation (ASF), Apache 2.0. The software controller is also available under commercial packages.

3.2.4 OpenDayLight (ODL)

In 2013, the OpenDayLight (ODL) project was launched as an open-source project to further enhance SDN [32]. It is hosted by the Linux Foundation with a framework supported by the industry and led by the community for the ODL controller. It was renamed as ODL platform, and opened to end-users and customers, providing a shared platform for those committed to SDN goals to find new solutions by working together. Apart from Beacon and Floodlight, ODL is also a controller based on the Open Service Gateway Initiative (OSGI) architecture. OSGI technology is an architecture for modular application development that allows the dynamic component system of Java. For instance, it supports applications running in the same network system as the controller. ODL is an open platform for developers that can be utilized by different projects and communities to contribute, add more features to it, and construct commercial-based products. It is a collaboration of vendors and projects to encourage innovation and a transparent approach. With its ability to support other open standards such as 12RS and NetConf besides the Open Flow protocol, it has added flexibility as its feature. Many SDN platforms such as Brocade are based on the ODL platform. ODL has the opportunity to progress further as it has extensive support from vendors such as Big Switch and Cisco. In addition to supporting the open protocol of OpenFlow, OpenDayLight also supports and operates other southbound protocols such as Border Gateway Protocol Link-State (BGP-LS) and Path Computation Element Protocol (PCEP).

3.2.5 Open Network Operating System (ONOS)

The ONOS is driven by Open Networking Lab (ON.Lab) as an open-source SDN. The project aims to construct an SDN operating system for communications service providers that has scalability, high performance and high availability. ONOS programs software or modules controlling and managing network components, including network elements such as switches and links. ONOS is adopted as a distributed system across multiple servers that provides fault tolerance while forming clusters for high resiliency. Thus, server failure or hardware or software upgrading will not impact operation. The ONOS kernel and its components are written in Java and deployed over Karaf OSGI container. Similar to ODL, OSGI components in Java ensure the modules in ONOS can work together and subsequently operate interchangeably in a single Java virtual machine. Furthermore, as it operates in the virtual machine, ONOS runs on several underlying OS platforms. ONOS was released as an open-source project on 5 December 2014, focusing primarily on communication service providers to be examined, evaluated, and improved by the SDN community [33,34].

4 Comparative Study of SDN Controllers

Every open-source controller provides manageable flow control that is able to practically deliver automated policy control through software software-defined network environment [26–28]. Table 2 provides a detailed comparative study between Nox/Pox, Ryu, Floodlight, OpenDayLight (ODL),

and Open Networking Operating System (ONOS), Table 3 describes the key features of ONOS while Table 4 provides a **c**omparison between different ONOS releases & the supported functions.

Table 2: Comparative matrix of SDN controller

	NOX/POX	RYU	Floodlight	OpenDaylight	ONOS
Original developer	Nicira networks	NTT	Big switch networks	Linux foundation	Open networking lab (ON.Lab)
First release	2008/2012	2012	2012	2014	2014
Language support	C/C++ & Python	Python	Java	Java	Java
OpenFlow support	✓	✓	✓	✓	✓
OpenFlow version supported	1	1.0, 1.2, 1.3, 1.4, 1.5, Nicira extensions	1.0, 1.2, 1.3, 1.4	1.0, 1.2, 1.3, 1.4, 1.5	1.0, 1.2, 1.3, 1.4, 1.5
Carrier Grade (Cluster)	–	–	–	–	✓
NFV Support		✓	✓	✓	✓
REST API	✓	✓	✓	✓	✓
Traffic Engineering		–		✓	✓
Load Balancing	✓	–	✓	✓	✓
Network Monitoring	–	–	✓	✓	✓
Web GUI	–	–	✓	✓	✓
NBI	✓	Openstack Neutron plug-in, REST	Openstack Neutron plug-in, REST	✓	✓
SBI	✓	OpenFlow, NET-CONF, OF-Config, OVSDB, XFlow	✓	OpenFlow, NET-CONF, OF-Config, OVSDB, LISP, PCEP, BGP, SNMP	OpenFlow, NET-CONF, YANG, OF-Config, OVSDB, LISP, PCEP, BGP, SNMP, P4, RESTCONF

(Continued)

Table 2 (continued)

	NOX/POX	RYU	Floodlight	OpenDaylight	ONOS
Documentation and learning curve	Low ✓	Medium ✓	High ✓	✓ High Well documented. Blogs, web portals, forums, subject groups, summits, tutorials.	✓High Well documented. Blogs, web portals, forums, subject groups, summits, tutorials.
Topology discovery Learning switch Network-wide switch	Discovery, Topology, Authenticator, Routing, Monitoring	–	–	–	Global Network Topology View, Authentication, LB, DPI, Routing, Monitoring, Intent Framework and Global Network View.
Network level	Enterprise	Enterprise	Enterprise	Enterprise (Data Center)	Service Provider (Telcos).

Table 3: ONOS key features

Features	Description
Distributed core	The SDN operating system working in a cluster is aimed to meet the carrier-grade requirement as follows (i) agility, (ii) resilience, (iii) high resiliency, (iv) high performance, (v) flexible scalability through multi-applications and capacity demands.

(Continued)

Table 3 (continued)

Features	Description
Northbound abstraction/APIs	The northbound interface allow a component in the network to connect and communicate with higher-level components. The interface is considered by some experts as the most critical since it will influence the applications that can be supported by the SDN. As the SDN needs to support a wide range of applications, some APIs will be placed in the network to cover the job, due to the ability and limitation of each API. Applications addressed by the northbound abstraction include management solutions, security and application orchestration across cloud resources. Southbound abstraction/APIs: A southbound API enables a component to communicate with a lower-level component such as physical and virtual switches, and routers. OpenFlow southbound interface which has outlined the standard protocol of SDN between the data plane and the forwarding plane. The interface helps the network to adapt itself to the real-time demands of business. Cisco and OpFlex are also southbound interfaces employed by users.

Table 4: Comparison between different ONOS release & the supported functions

Version	ONOS release	Core	Northbound	Application/Use Cases/GUI	Southbound
1.12.0	Magpie	Improved failure detection for faster mastership election times Migration of all distributed primitives to Atomix	gRPC Brigade Added	New Use Cases Simple Fabric: Intents-based leaf-spine fabric application for physical switches. GUI Brigade Application View filter	Packet Optical Additional device support by Equinix, Polatis, OpLink
		Raft log compaction optimizations Offline backup/restore	LinkService HostService MeterService ComponentConfig Service	Added Details Panel for Ports, Meters, and Groups views Network Virtualization Brigade Virtualization core-distributed virtual flow rule store added	Support for PowerConfig API on Polatis device REST API to allow applications outside ONOS to access PowerConfig API Ciena transponder supports more API/Behaviours & protocol options
			RegionService ApplicationService Support gRPC service registration with a static binding mechanism	OFAgent handles PORT_MOD, FLOW_MOD, GROUP_MOD, METER_MOD messages Log filtering per OFA- gent tenant capability is now available	

(Continued)

Table 4 (continued)

Version	ONOS release	Core	Northbound	Application/Use Cases/GUI	Southbound
1.11.0	Loon	Re-architected Raft implementation Per-primitive sessions Simplified log compaction Multi-version serialization Performance tests added Fault injection/linearizability verification tests added New flow rule store Backed by partitioned consistent primitives	gRPC Brigade (ONF, ZTE, Inspur) First services merged Structure for gRPC system and naming conventions established Translation system improved	GUI Brigade (Villa-Tech, ONF) Added new developer tools for GUI development NPM + Gulp Linting Code Coverage Bundled JS and CSS Improved load performance by reducing the number of requests made Added a few more GUI diagnostics tools, such as port filtering + 19% code coverage from tests Ported some Topology functionality to Topology 2 Updates to the Flow View Enable table column reordering Show Application ID	Dynamic Configuration Brigade (Huawei, NEC, ONF) P4 Brigade (ONF, POSTECH, NCTU, Barefoot, AT&T)
1.10.0	Kingfisher	Virtualization improvements Support of flow objective and mastership services for virtual networks Performance improvements	RESTCONF adapter for YANG Â· Intent Framework improvements Intent framework and VPLS support for OFDPA pipeline (e.g., Accton switches) New intent installer to let users create their intent installer logic and plug them in at run-time Domain intents: extension of Intent Framework to generate domain intents Route subsystem enhanced to support multiple routes per prefix	GUI Added packets/second mode to traffic monitoring Dark theme reinstated Regions/layouts enhancements-layouts can use grid or geo backgrounds for example Packet Optical OpenROADM device, network, and service support Full protection behavior incl. hardware-based demonstration Datacenter/SONA Support for VLAN-based virtual network provisioning Unit tests and support of OpenStack Tempest API vRouter Enable multiple FPM connections for HA Added support for multiple control plane redi rects for multiple Quagga routers	NETCONF adapter for YANG Apache Mina library for NETCONF OpenFlow 1.4 support LISP subsystem enhancements for mapping services

(Continued)

Table 4 (continued)

Version	ONOS release	Core	Northbound	Application/Use Cases/GUI	Southbound
1.9.0	Junco	Fabric Enhancements, IPv6 segment routing, Virtual Private Wire Service (VPWS) Framework HA enhancements Kafka now uses HA primitives Virtualization support (to virtualize ONOS instances)	Optional Guaran-teed Bandwidth Allocation Protection Specification Shared resource modeling Hashing support for ECMP traffic distribution	Scalability improvements Regionalization support vRouter IPv6 support	YANG Models. Enhance YANG Compiler for auto-generation of JAVA as per YANG. YANG Runtime for automation of CODEC. XML/JSON Serialializers to adapt Models. Dataplane Enhancements New TL1 southbound for Lumentum WaveReady
1.7.0	Hummingbird	–	TE Topology NBI Implement RES-CONF client and server	–	–
1.4.0	Emu	Multicast, flowrule store persistence	Resource reservation, gRPC, security, iperf	New overlays, samples, GUI for archetypes, doc	NetConf/Yang, optical support
1.3.0	Drake	TLS for security, metering, multicast	Login authentication, basic virtualization	Topology overlays, link highlighting	OVSDB, PCEP, VXLAN, NetConf/Yang
1.2.0	Cardinal	Dropped Hazel-cast, dist primitives	MPLS+tunnel intents, performance Flow objectives	Multiple GUI view	TL-1, NetConf
1.2.1					
1.2.2					
1.1.0	Blackbird	Events, RAFT maps	AIF, clustering app deployment	Modularized GUI	Multi-table Open-Flow
1.0.0	Avocet	Distributed state, management for scale, performance, HA	Basic App Intent Framework (AIF)	Basic GUI component for developer and user	OpenFlow support

4.1 Origin and Community Support

The first Open Flow SDN controller was developed by Martin Casado, Nick McKeown and Scottt Shenker under a project at Nicira Networks in 2007. The main focus of Nicira was on SDN and network virtualization. NOX has been the foundation for numerous research projects in the early growth of SDN research ever since it was made known to the research community in 2008. POX, a NOX-based controller, supports Python together with C++. Ryu, an open-source SDN framework, was developed and supported by NTT Labs. Similar to other open-source controllers, RYU also enables the community to develop management and network applications with the existence of its well-defined APIs. As one of the earliest SDN controllers, Ryu has been widely used by various stakeholders from universities and industries including SDN application developers, network device developers, and network service providers. Apart from being employed by the NTT cloud data centre, the NSA is also utilizing the Ryu SDN controller to track and control its network. Floodlight is a Beacon-based OpenFlow controller supported and developed by a community of open-source developers and engineers from Big Switch Networks Inc. that initially began at Stanford University and UC Berkeley,

released under an Apache 2.0 license in January 2012, Floodlight is freely downloadable to catalyse third-party application development around it.

In April 2013, OpenDayLight was announced by the Linux Foundation as an open-source project that encourages openness of the SDN platform and enables external application design and development. The OpenDayLight Project was developed by engineers from Cisco, HP, Juniper Networks, IBM, RedHat, Microsoft, Ericson, and VMWare. On 5 December, 2014, the Open Networking Lab ON.Lab, AT&T and NTT Communications together with their industry partners released the ONOS source code to initiate the open source community. ONOS later joined the Linux Foundation, as with most open source projects, has a GitHub page where collaborators can contribute changes to the code. The ONOS community has seen a steady growth in membership which now includes developers and users from Google, AT&T, Verizon, NTT, Ciena, Fujitsu, and Huawei. ONOS is distributed under the Apache 2.0 license. Tables 4 and 5 compare between different ONOS releases and their supported functions.

Table 5: ONOS core

Release	Remarks
Avocet	First release carrier-grade features (distributed architecture, scalability, high availability and high-performance features).
Blackbird	Synchronization and redundant state across a collection of devices were maintained by including an additional RAFT and network mapping capabilities were more consistent.
Cardinal	The core was strengthened. Primitives and APIs were created to facilitate other services and apps to utilize the distributed state management in order to meet carrier-grade requirement.
Drake	This release provides IGMP snooping with PIM-SSM and multicast for-ward app, security enhancement of TLS, and OpenFlow QoS meter support to collect devices or port information.

4.2 Architecture

NOX/POX architecture consists of two main components: core and sub-components. The core component provides helper methods such as network packet process, threading, and event engine. It, also, supports the OpenFlow APIs for interaction with OpenFlow switches and I/O operations support. The in-built component of NOX makes up the middle layer. The connection manager, event dispatcher, and OpenFlow manager are self-explanatory. The directory structure is examined by the dynamic shared object (DSO) deployer for any components being positioned as DSOs. Cooperating components generally make up the NOX applications that provide the required functionality. In brief, a component captures the specific functionality that is accessible to NOX. RYU, the component-based SDN, provides software components with well-defined northbound Python APIs and supports southbound OpenFlow protocol.

The core architecture of Floodlight is modular and includes topology, device and web management, path computation, OpenFlow counter store, and state storage abstraction. These controller components load as a service with an interface that exports state, and provides extensible REST APIs and event notification systems through applications. OpenDayLight architecture is constructed based on the Open Services Gateway Initiative (OSGi), a modular development framework where the entire

platform is constructed of modules that are loosely coupled. OpenDayLight layered architecture is defined by clear integration points and APIs, allowing end users and networking vendors to explore the powerful SDN potential of ODL.

Networking technologies and hardware from diverse vendors can be leveraged using ODL as ensured by the southbound interface to ODL. On the other hand, end users and other cloud technologies such as OpenStack are provided with APIs by the northbound interface. ONOS architecture tiers are constructed of the core layer, northbound abstraction and southbound abstraction layer. It was specifically created as a response to carrier-grade networks' demands of performance, high availability, and scalability, with well-defined abstractions.

4.3 OpenFlow Protocol Support

Most of the SDN controllers in the market employ the OpenFlow protocol. The protocol is applied by the SDN to configure network devices and select the best path for the data packets to follow. NOX is the original OpenFlow controller that can control and manage OpenFlow devices in the network. It includes sample applications that are written in Python and C++. Ryu is another open-sourced controller that supports OpenFlow protocol. The Floodlight controller which is constructed based on the Beacon controller was released by Big Switch and aims to be employed by production enterprise networks. Built on the OSGI framework, Beacon is one of the earliest Java-based OpenFlow controllers. Applications developed on this platform can be regulated at run-time without disconnecting switches. The OpenDaylight controller (ODL) which the Linux Foundation hosts supports the OpenFlow protocol besides supporting other open SDN standards. ONOS supports multiple southbound protocols including OpenFlow 1.0. Earlier ONOS versions supported the basic functionalities of OpenFlow which was only a southbound protocol at that time.

4.4 Programming Language

The programming language employed by NOX is C++ while POX uses C++ and Python. Floodlight, OpenDayLight, and ONOS are Java-based while RYU is written in Python.

4.5 Operating Support System

The Linux operating system is applied by NOX/POX, RYU, Floodlight, OpenDayLight and ONOS. The Java-based project spearheaded by the Linux Foundation aims to accelerate the adoption of SDNs such as ODL and ONOS. The open-source Linux-based SDN can offer full control, and software support with extra tools for advancements and deployment.

4.6 NFV Support

Network Function Virtualization (NFV) aims to enhance the software-based networking approach similar to SDN. Floodlight, OpenDayLight, and ONOS are open-source controllers that are compatible with the NFV platform. GS NFV-EVE 005 report has called for the positioning of SDN controllers in the ETSI NFV architectural framework. Project CORD (Central Office Re-architected as a Datacentre) has merged NFV and ONOS SDN controllers.

4.7 Carrier-Grade Support

ONOS is the first SDN controller that meets carrier-grade requirements such as throughput, availability, and scalability. ONOS is the core engine of many commercial-based SDN controllers. ONOS carrier-grade controller is motivated by the high performance, scalability, and availability

requirements. Driven by ON.Lab with a healthy community of users and vendors that work together to develop, deploy, and support the platform. Despite the new open-source controller, ONOS documentation is well-documented and regularly updated with new information. A vast range of instruments to develop a new application based on the controller are provided by ONOS. Like any other controller ONOS support OpenFlow versions 1.0, 1.2, 1.3, 1.4, and 1.5.

To communicate with network infrastructure, ONOS supports multiple southbound protocols including Open-Flow, NetConf, and SNMP. By exposing northbound APIs and unique service abstraction of the application intent framework, ONOS makes it open for the needs of multi-use-cases services, application development, policy injection and other operations. In ONOS, unlike other controllers, a distributed architecture is designed for clustering targets for scalability, HA, and performance, this is to meet mission-critical service providers' network demands. ONOS is globally deployed in Research and Educational Networks (NRENs) [35,36]. Service providers like Huawei deployed ONOS in the China Unicom network [37]. In AT&T, ONOS was delivered as part of the deployment of Central Office Re-architected as Data Center (CORD) and Mobile-CORD solution [38].

5 Open Networking Operating System (ONOS)

ONOS is a network operating system that was specially built for service providers. The mission-critical and complex nature of the service provider networks requires a control platform that is highly reliable, scalable, and has proven capabilities. It provides well-defined network abstraction and APIs to enable application development and interaction with network elements. Furthermore, it enables operators or service providers to develop and build production SDN solutions. Thus, ONOS offers an open-source network operating system, and carrier-grade SDN control plane [33,34].

5.1 ONOS Evolution

On 5 December, 2014, ONOS launched its first open-source release known as Avocet version 1.0.0. Releases and development of ONOS follow a time-based release cycle and a new release has been issued every quarter of the year. Blackbird version 1.1.0, which is the second release of its open-source SDN controller, was introduced on 17 March, 2015. Its main goal was to improve the performance and scalability characteristics of ONOS, specifically in the areas of intent flow operation throughput and latency. This release was developed with several performance improvement targets which included subsystems, APIs, cluster application development and management, enhanced map functionality based on RAFT, device driver framework, GUI interface, and IPv6 support. With enhanced performance, ONOS is able to achieve the high target of million flow operations of less than 10 ms latency while maintaining a diversified range of operations and network configurations through intents and flow rules. This performance study demonstrated how ONOS open-source SDN controller can be scaled out as required to deliver high network resiliency while maintaining high performance and availability.

ONOS has seen thirteen major releases and thirty different versions since its first release in 2014. Over four years later a lot has changed, ONOS open source SDN solution has gained many new features and functions in its core, interfaces level, subsystem, applications, API, and devices. Each ONOS release made a number of significant advancements and optimizations in many areas, the performance and scalability elements have continued improvement in subsequent releases. Furthermore, the latest Magpie release version 1.12.0 shows that ONOS can react to a topology event latency for example switch failure in less than 10 ms compared to 10.4 ms in the case of Loon release, and the

intent operations performance has increased from 3K events/s in Loon to 5K events/s in Magpie while maintaining 3 million flow operations per second. The details are shown in Tables 4 and 5.

The growth of ONOS development and continued enhancement of the project has seen significant momentum in platform features. The iterations of ONOS release from the first release of Avocet till the latest Magpie. A comparison of four ONOS releases, Avocet, Blackbird, Cardinal and Drake are provided in Table 5.

5.1.1 Core

Use cases as specified by service providers in supporting core architecture, new applications, and Proof of Concepts.

5.1.2 Northbound

The northbound interface refers to the API's ability of an ONOS controller to create communication with the application layer and coordinate to program the network and request service from it. These are some of the values of SDN controller northbound APIs that are able to support various applications. These APIs are also able to establish connectivity with the NFV platform, for instance, OpenStack or any cloud management service. The most popular northbound interface protocol is the REST protocol. To enable interface interaction to deliver service deployment. This involved interface with other protocols, and applications, and improved performance. Four improvements of the ONOS Northbound API are compared in Table 6.

Table 6: ONOS Northbound API

Release	Remarks
Avocet	Build with Application Intent Framework (AIF) subsystem for global policy controls which enables applications to define it requires resources and complete sys tem tasks.
Blackbird	The intent framework for ONOS was extended and upgraded with more cluster features and application deployment capabilities. Improved performance with new metrics SDN test methods.
Cardinal	High performance was delivered by (i) MPLS and (ii) enhancement tunnel intents support, network conflict resolution capabilities, and flow-objective subsystem in the distributed core for device-agnostic SDN deployment.
Drake	Login authentication and virtualization basics were added. To improve and simplify the setup of cross-domain constructs or flows, a new northbound intent abstraction was introduced. This also simplifies specific domains with an intent.
Emu	The enhancement continues in resource reservation API and supports a new type of resource, integration of Yangforge for Yang-based model development. Emu release also supports gRPC, dynamic REST interfaces, and bandwidth meters mechanism (SBI). Additionally, it also supports RIB multicast APIs and new intent domains (IETF's Service Function Chaining) protocols.

5.1.3 Southbound

The southbound interface is used to communicate and enable programmable control according to real-time demands and requirements between the ONOS and network elements over the network.

Changes to forwarding rules in the data plane were made possible through exposed APIs of the network elements. The network elements could be routers, switches, servers, etc. The most popular SDN protocol for southbound APIs is OpenFlow. Other southbound interface supporters of ONOS are NetConf and OVSDB. L3 protocols of OSPF, BGP, and IS-IS which serve as south-bound interfaces are introduced to support hybrid networks or to maintain traditional networking [30]. Southbound capabilities include (i) command and protocol interface competencies supporting other open-source distributions, (ii) protocols, (iii) standards and applications as well as security, (iv) OPNFV, (v) OpenStack, and (vi) the ability to model external traffic and new services. A summary of ONOS Southbound API is provided in Table 7.

Table 7: ONOS Southbound API

Release	Remarks
Avocet	ONOS avocet release supports OpenFlow version 1.3 (OF 1.3). The first release also supports other SBP protocols (via plugins).
Blackbird	"Multi-table support is allowing capabilities to handle IPv6 traffic, enhance multi-path and segregation of logical events while improving table pipeline in production ASICs based networking."
Cardinal	"Additional SBI added in this release includes TL1, PCEO, and NetConf. This newly added interface is also to support solution POCs."
Drake	"Drake releases support path computation and more functionalities in PCEP rules. TLI networking API is further refined by adding OVSDB, VXLAN, and device configuration capabilities."
Emu	Topology information from the network was able to be collected and made available to other apps by adding a new plugin to the ONOS controller (Border Gateway Protocol with Link State Distribution extension (BGP-LS)). Further introduction of ECI in ONOS controller improves optical applications of supporting ODU multiplexing and cross-connect services through SBI based on ONF standards.

5.1.4 Graphical User Interface (GUI)

ONOS GUI focuses on the ability to visualize the network and device's state together with other information such as traffic flows and policies. Different releases related to ONOS GUI are summarized in Table 8. ONOS GUI includes:

Table 8: ONOS GUI

Release	Remarks
Avocet	Provide a single-view GUI focusing on the network topology and basic end-to-end flow/traffic information.

(Continued)

Table 8 (continued)

Release	Remarks
Blackbird	"Upgrade user interphase (UI) framework to AngularJS framework. Enhancement of topology view with the capability to display additional port numbers on highlighted links. Furthermore, a device view and information were added in the Blackbird release. To provide more UI content in ONOS, a UI extension mechanism was applied to have the capabilities to enrich content into the UI."
Cardinal	"All the POCs (Multi-layer Network Control (MLNC), the SDN-IP peering ap-plication, Internet2's real-world deployment showcase, and CORD and VNFs (vCPE, vOLT, vBNG) at the central office) at ONS2015 were supported by GUI. Information on the hosts, links, intents, applications, and instances are also populated as flows, ports, and groups per device. The new toolbar was supported in the topology view."
Drake	"Topology overlays were included to allow apps to highlight links programmatically and tailor the content of the Summary and Details panels of the Topology View. With Drake, ONOS' GUI interfaces became secured by default."
Emu	And added new archetypes for views and topology overlays. Topology overlays provide the capabilities for the users to customize the view of the network. The sample application was also added to demonstrate the content insertion method of UI contents.

a) ONOS Architecture

Build to meet carrier-grade requirements, ONOS architected for scalability, high availability, and performance. It presents innovative abstractions of northbound and southbound interfaces. It is a module that allows well-defined software functions, abstractions, and customization. Fig. 6 shows the tiers architecture of ONOS.

ONOS is a distributed system and runs as a cluster. Symmetrical instances of cluster deployed as a service on a server enable failure in the event of an ONOS instance breakdown. As the requirement grows, control plane capacity can be scaled out without network disruption. The created instances form and operate in cluster mode. Whether it runs on single or multiple instances, ONOS will be seen as a single platform. The distributed core manages the states across instances and it is the key component enabling scale and high availability. Table 9 summarizes the characteristics of ONOS.

A pluggable Southbound allows OpenFlow or other protocols to interact with network elements. Southbound abstraction is composed of network elements including switches, hosts and links generally known as an ONOS object. Through the abstraction of the network element, the addition of devices and protocols such as NetConf or OpenFlow can be materialized. The Southbound abstraction in ONOS allows plugins of various protocols and devices. Thus, it enables ONOS to control and manage different types of devices and protocols.

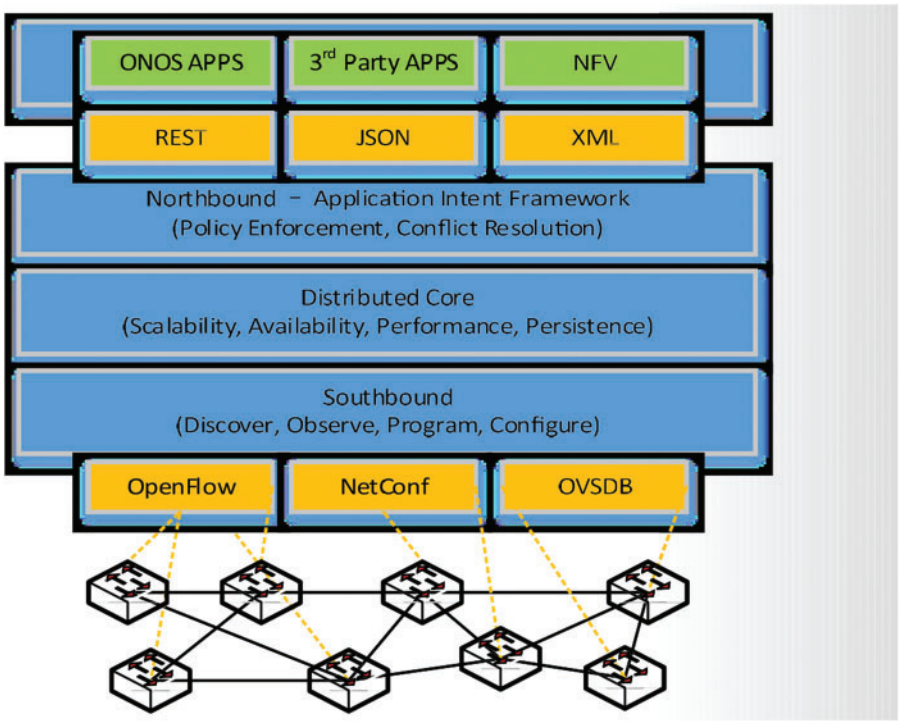

Figure 6: ONOS Architecture tiers

Table 9: Characteristics of ONOS

Characteristic	Notes
Distributed Core	"Carrier-grade features provide high availability, scalability, and performance. Exposed agnostic API protocol for applications and providers layer. Able to perform as a cluster based and replication add agility to the SDN control plane."
Northbound abstraction/APIs	NB API for applications to interact with the network model.
Southbound abstraction/APIs	SB API for providers to inject environment data and to receive control inputs
Software Modularity	Allows customizations and integration to minimize complexity and offer a stable platform for evolution.
Drake	"Drake releases support path computation and more functionalities in PCEP rules. TLI networking API is further refined by adding OVSDB, VXLAN, and device configuration capabilities."
Emu	Topology information from the network was able to be collected and made available to other apps by adding a new plugin to the ONOS controller (Border Gateway Protocol with Link State Distribution extension (BGP-LS)). Further introduction of ECI in ONOS controller improves optical applications of supporting ODU multiplexing and cross-connect services through SBI based on ONF standards.

Northbound abstractions and APIs are an application programming interface. The global network view is physically distributed across multiple controllers but logically centralized across multiple controllers. It allows applications to program their view of the network while users do not need to know the details of how the service will be established. Thus, it enables network programming at a high level.

Software modularity architecture is designed to ease software maintenance. The software is constructed into modules and interrelated with each other. As a development paradigm, modular software emphasizes self-contained, flexible, and independent pieces of functionality.

This feature allows new functions to be added whenever desired, and unwanted functions to be removed, making it a user-friendly software that has great flexibility. The diagram in Fig. 7 shows that the main structures of ONOS consist of tiers centered around the distributed core. Thus, the NBIs and SBIs at the macro level provide an initiative to insulate structures from each other. In addition, the independent architecture is open for any new application or new protocol adapters or plugins to be added as required. In conclusion, with the level of abstractions and open APIs available, ONOS offers an open SDN solution that enables the innovation, and development of enriched network management applications and next-generation services which is compatible with legacy network elements and existing network protocols. This is also encouraging the transition from static network infrastructure to automated network control.

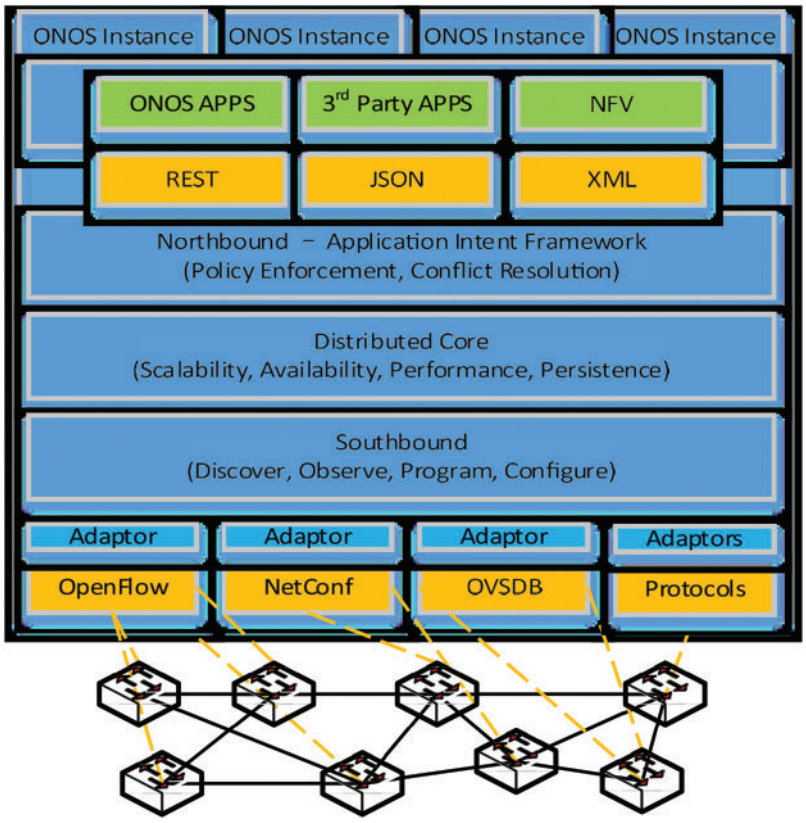

Figure 7: ONOS Architecture tiers

b) ONOS Application Deployed

Open source is one of the spurring factors behind the ONOS deployments. Numerous groups of users which include network developers, researchers to industrial network architects have improvised and utilized different ONOS- based applications to meet their specific requirements. Table 10 lists below the different applications developed and utilized based on the ONOS controller.

Table 10: Application deployment based on ONOS

No	App name	Description	Developed by
1	Inter-Cluster ONOS Network Application	Manage the intercommunication of geographically distributed ONOS clusters and deliver faster controller re-response time during network events such as failures or congested links.	CREATE-NET and the University of Rome Tor Vergata
	ICONA [36,39]	ICONA project.	
2	SDX L2/L3 [40]	Manage Internet eXchange Points (IXP) via L2 and L3 connectivity. SDX-L2 ONOS application.	CNIT, CREATE-NET, GEANT and the University of Rome Tor Vergata
3	Mobile-CORD [41]	A platform for 5G mobility exploration based on commodity HW and open source solutions. M-CORD ONOS application.	OpenCord, Cavium, Radisys, Airhop, Nec, Cobham, Intel, Tech-Mahindra, Viavi, Netronome, and Lime-Micro
4	Virtual Private LAN Service (VPLS) [34]	Creates Virtual Private Broadcast L2 networks based on VLANs, on demand. VPLS ONOS application.	ON.Lab, AmLight, NCTU
5	Transport SDN TSDN [42,38]	SDN-based IP Optical network transport with bandwidth on demand (BoD), network virtualization, and VTS features.	Huawei and ONF
6	Transport SDN TSDN [43]	SDN-based Carrier Ethernet network with BoD and OSS/BSS	TM R&D and Fiber-Home
7	vRouter [44]	Emulate hardware-based. Layer 3 IP routing. The application was created in the form of a VNF.	Tech-Mahindra
8	DHCP Relay [44]	Agent to insert information about client's identity into DHCP client request being sent to a DHCP server. The app modifies DHCP packet sent to the server.	Tech-Mahindra
9	Castor [45]	Provides L2/L3 connectivity for SDX.	AARNET

(Continued)

Table 10 (continued)

No	App name	Description	Developed by
10	SDN-IP [46,47]	SDN-based IP transit network using BGP protocol and L3 connectivity with other legacy devices while allowing multiple Administrative Domains to communicate through the SDN network. SDN-IP ONOS application.	AmLight Internet2 KREONET and NCTU
11	SD-WAN [48,49]		KREONET-S

As stated earlier the architecture of ONOS is based on three key pillars; (i) scalability, (ii) high availability, and (iii) high performance. Its well-known API abstractions and module-based architecture are remarkable features that encourage creative innovations. The innovative features of ONOS simplify the creation, deployment, management, and control of applications. Above application, use-cases simulate and deploy based on ONOS SDN controller and the features to meet service provider carrier-grade level.

(i) SDN-IP: Fig. 8 shows how the cross-regional testbed was first deployed between 2015 and 2016. The deployment testbed covers five continents involving thirteen Research and Education Networks (RENs) and Research Institutions. Initiated by AMLight, GEANT, and Internet2, this testing facility is then linked to several RENs from Asia, Europe, and America [46,50].

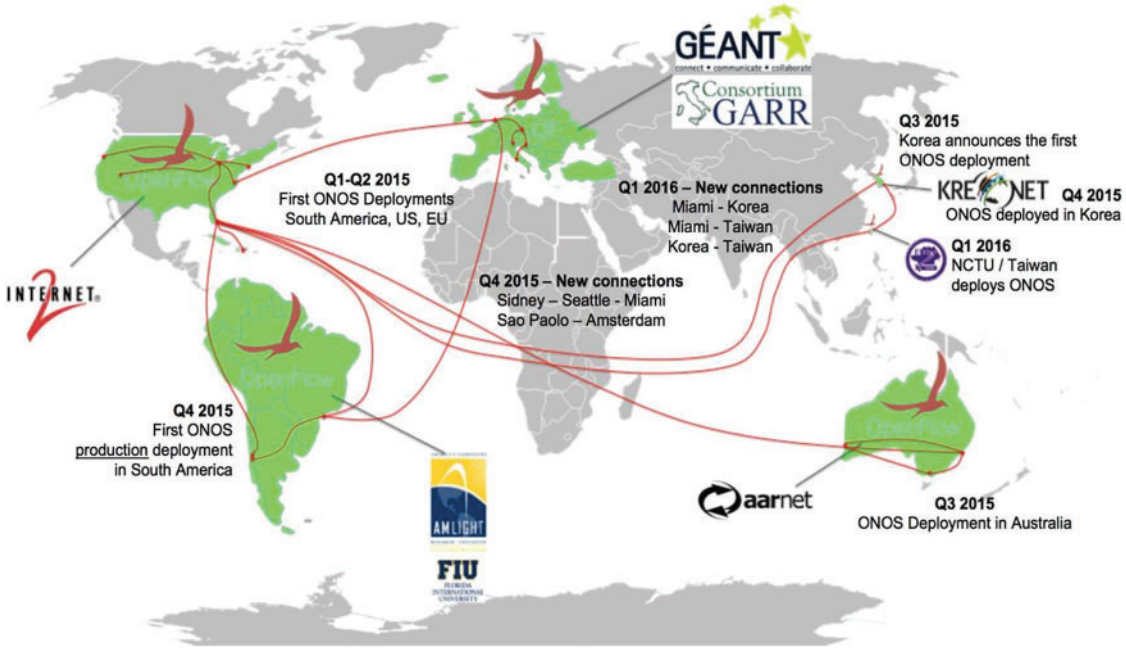

Figure 8: SDN-IP global deployment map [50]

AmLight decided to join the SDN Global Deployment with several considerations: (1) create a global SDN network; (2) provide L2 and L3 connectivity without legacy equipment in the network core; (3) bring network innovation by exploiting new applications developed internally at AmLight. It is also worth highlighting that as the network aims to become a platform for innovation, both CSIRO/AARNet and GEANT have developed and employed their SDN/IP application to bridge the legacy IP/BGP and the SDN worlds. This accomplishment highlights the flexibility of ONOS in accepting a new software piece and making it interoperable with existing applications [50].

As a result of the project led by ONOS, three regional networks in the US, Latin America, and Europe have emerged and linked to each other. The project aims to create a global SDN network that will allow entities to communicate at Level 3 without legacy routers in the network core. The project also aims to show that ONOS can function to provide high performance, high availability, and scalability in real networks. In making these goals a reality, a network of partners have utilized and installed ONOS, provided feedback, and then deployed the latest version of ONOS using an agile deployment model. In its current state, three SDN networks have joined together to form the network.

Internet2 is a collaboration between researchers and educators from the universities, government, and industries to develop advanced Internet technology. The Internet2 is seen as a 'playground' in testing network capabilities in a real-world environment. In 2015, Internet2 and Internet2 NOC at Indiana University migrated to an SDN-based network with an ONOS-based solution. Five universities in North America: The University of Utah; Duke University; the University of Maryland; the Indiana Gigapop; and Florida International University in Miami were connected to the virtual slice applying ONOS. The ONOS and SDN-IP network abstractions are achieved by using a Flowspace firewall that allows university legacy routers to be connected to Internet2 through OpenFlows switches. The deployment connects South America, Europe, and the United States. In South America, the AM Light network connects Florida International University in Miami with REUNA and RedClara in Santiago, Chile; ANSP and RNP in Sao Paulo, Brazil; and CKLN in the Caribbean. The network consists of six OpenFlow switches and utilizes Brocade and Juniper physical hardware. Internet2 and AM Light are connected through a legacy router at Florida International University in Miami. Gèant/GARR, in Europe connects Universitá Roma Tor Vergata in Rome and CREATE-NET in Trento, Italy over five OpenFlow switches [47,50].

In total, there are more than 50 OpenFlow switches connecting 14 institutions on three continents in this project. The project has successfully validated ONOS applications such as SDN-IP, SDX-L2/L3, and Castor. However, ONOS has suggested to providers to rethink Open Flow support and consider multi-table pipelines. The mentality of R&E network operators have to change to software and agile methodologies and more R&E network operators are needed to focus on stability, performance, and scalability. In the meantime, work on commercial deployments is in progress at Kreonet in South Korea and AARNet in Australia.

(ii) Data Centre SDN: As shown in Fig. 9, Multilayer SDN Control of Packet and Optical Networks. This demo presents the integration of SDN control of packet and optical networks with ONOS to maximize the full potential of optical network agility. This demo demonstrates the effectiveness of ONOS SDN control across devices and technologies involving optical devices from optical vendors such as Ciena, Huawei, and Fujitsu. A logical overlay is placed over the ROADM network and the networks are coordinated at the optical by treating both layers logically using a single SDN layer to control. As a result, interoperation between the optical layer and the IP layer is achieved [50].

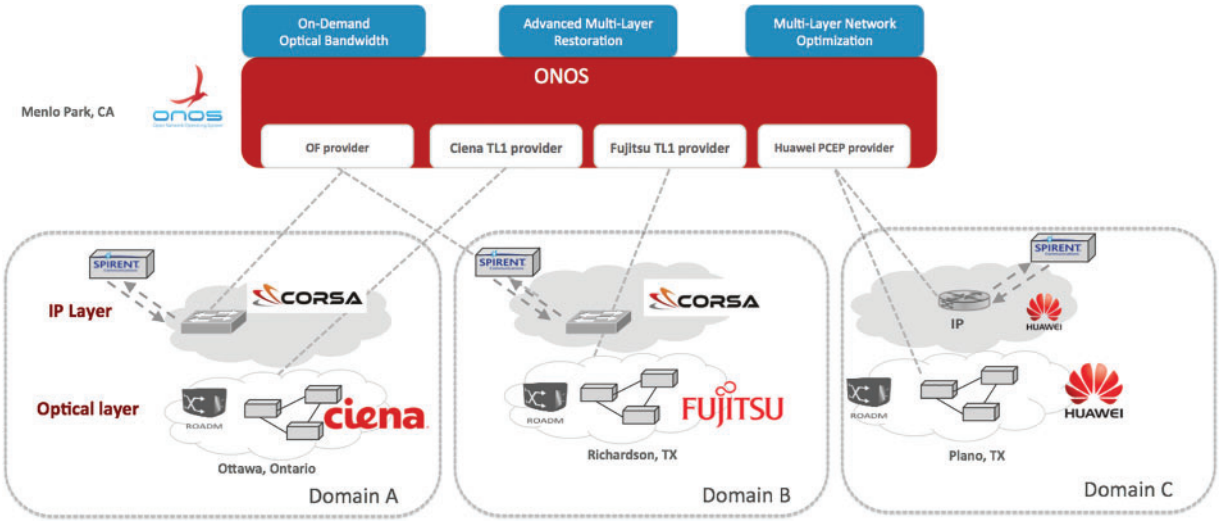

Figure 9: SDN-DC global deployment [50]

These demonstrations simulate the real environment and how ONOS operates and reacts in a production environment. In this simulation, a single control plane supports a multi-vendor and multi-protocol network which is represented by a virtual machine acting as a switch and modified mininet. This demonstration shows how multi-layer SDN control can be achieved by ONOS controller regardless of the diversity of applications and vendors involved.

(iii) SDN Network Function Virtualization (NFV): Argela research and development group of Turk Telekom deployed SDN NFV infrastructure and adopted ONOS as a network operating system (see Fig. 10 which describes Argela's SDN-NFV framework). Argela has initiated the deployment of ONOS at three different Turkish government institutions besides Turk Telekom's intranet. The infrastructure provides extensive centralized analytics, policy management, zero-touch traffic management, enhanced topology management, and complete network security.

Configured ONOS allows system functions to cover multiple aspects of user access security, data, system control and application layers. This includes security aspects of DDoS threats, IDS and network access protection. Ever since its emergence, Argela has a wide experience with SDN controllers, integrating SDN controllers with Open Stack and even developing their controller known as YakamOs. Their experience and commitment to the research and development in wireless communication networks have earned them a good reputation as a well-known high technology company [50].

c) ONOS Based Commercial Solution and Project

In addition to being deployed in a testbed environment and experiment, ONOS was materialized by telco vendors and technology providers as a new SDN project and some of them were developed as commercial products. By leveraging the open-source ONOS SDN controller, new products and projects were introduced. Vendor-based solutions on ONOS are shown in Table 11.

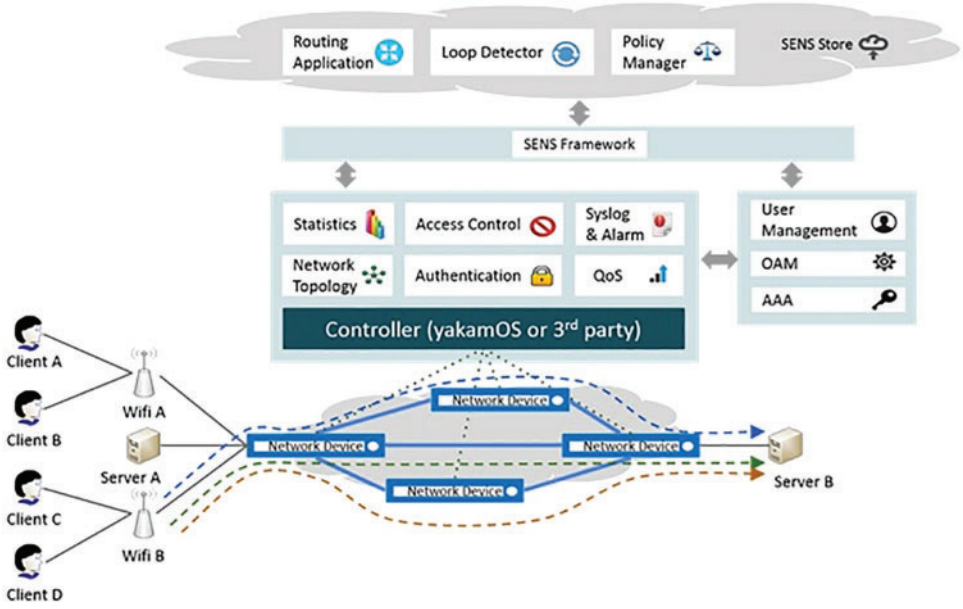

Figure 10: Argela's SDN-NFV framework [51]

Some other vendors have already announced and plan to build commercial products and solutions based on ONOS and other vendors are planning the same activities. SK Telecom, Turk Telekom, KT, Samsung, Telefonica and many more are some of the players that have deployed ONOS-based solutions for several use cases and commercial products. ONOS is seen as a critical element of many SDN projects as the SDN controller [52].

Table 11: Product solution and project deployment based on ONOS

No.	Vendor/Provider	Solution	Description
1	Huawei [53,54]	Agile Controller 3.0 (AC 3.0)	Introduced in Huawei Connect 2016, AC 3.0 main target on the enterprise campus, data center networks, WAN, and IoT. Improve the resource scheduling, network efficiency for carrier customers, and customer experience.
2	Ciena [55]	Blue Planet ONOS	A commercial grade SDN controller focused on the data center use case, multi-domain and orchestration.

(Continued)

Table 11 (continued)

No.	Vendor/Provider	Solution	Description
3	NTT*	Open and Disaggregated Transport Networks	To build data center interconnect using disaggregated optical equipment, open and com mon standards, and open source software.
	T&T, SK Telecom, Verizon, China Unicom and NTT [41,56]		
4	TM R&D Fiberhome [44]	Transport SDN	Providing bandwidth on demand over packet transport network. Customers are able to manage and control bandwidth requirements through BoD portal.
5	ECI [57]	SmartLIGHT	Focusing on multilayer control of IP and optical layer networks. ECI developed its own ONOS-based SDN controller to be part of the SmartLIGHT solution.
6	vRouter [45]	Emulate hardware-based Layer 3 IP routing. The application was created in the form of a VNF of the NFV platform	Tech-Mahindra
7	DHCP Relay [45]	Agent to insert information about client's identity into DHCP client request being sent to a DHCP server. The app modifies the DHCP packet sent to the server.	Tech-Mahindra
8	Castor [46]	Provides L2/L3 connectivity for SDX	AARNET

(Continued)

Table 11 (continued)

No.	Vendor/Provider	Solution	Description
9	SDN-IP [47,48]	SDN-based IP transit network using BGP protocol and L3 connectivity with other legacy devices while allowing multiple Administrative Domains to communicate through the SDN network. SDN-IP ONOS application,	AmLight Internet2 KREONET and NCTU

Note: *https://group.ntt/en/ (accessed on 10/01/2024).

6 Advances in the Investigation of Open SDN Controllers

In the literature, several research works have been carried out extensively on various topics of SDN such as comparison studies, controller performance and evaluation, SDN models and architecture, traffic engineering, security, adoption, deployment, and development applications of SDN controllers. In this section, several related studies and some of their drawbacks will be briefly discussed.

6.1 Performance Evaluation, Comparison and Analysis

Tootoonchian et al. [58] are considered among the pioneers in conducting a comparative analysis of SDN controllers. The main focus was controller performance by NOX-MT, Beacon, and Maestro which was rather limited. However, with the rapid advancements in the performance of controllers, NOX-MT, Beacon and Maestro have been surpassed by POX, Ryu, FloodLight, OpenDaylight, and ONOS. The underlying review in [59] outlined a comparison between five controllers that includes TLS support, open interface, Graphical User Interface (GUI), Representational State Transfer (RESTful) API, documentation, modularity, platform support, virtualization, OpenFlow protocol support, and OpenStack network support known as Neutron. The Multi-Criteria Decision Making (MCDM) method named Analytical Hierarchy Process (AHP) adopted a monotonic interpolation or extrapolation mechanism to analyze the comparison. In this method, the values of the properties were mapped to a value on a pre-defined scale. Results from the adopted AHP in the comparison concluded that "Ryu" was the best controller based on their requirements.

An advanced study of SDN/OpenFlow Controllers was carried out in [27,60] by employing a framework known as hcprobe. The main objective of this study is to test and analyze the efficiency of SDN controllers such as NOX, POX, Beacon, Floodlight, MUL, Maestro, and Ryu. The study concluded that maximum throughput could be achieved by Beacon and most of the controllers were capable of coping with an average workload during long-term testing. However, there were possible security issues with the tested controllers.

In [61], the proactive and reactive operating modes were compared. They reported that the proactive mode performed better than the reactive mode since the rules in the proactive mode were loaded to the switch in the beginning. On the other hand, the rules in the reactive mode were loaded to the switch each time it received a packet with no matching rule in its flow table. Another study conducted in [62] suggested further considerations when devising a new controller. Two types of architecture were considered: First: static partitioning with static batching. Second, a shared queue

with adaptive batching. SDN controllers tested were Beacon, Maestro, NOX-MT, and Floodlight. The highest performance was recorded by Beacon which employs static batching while Maestro, which utilizes adaptive batching, presented the best latency records.

Studies carried out by [61,63] reported that the choice of the programming language has a great impact on the mobility and performance of the controller. Java was selected as the best programming language since it supports multithreading and is a cross-platform language. On the other hand, Python had issues with multithreading on the performance level, and C, and C++ encountered problems with memory management and the Net languages were dependent on the runtime platform; compatibility with Linux is not supported. Thus, Java-based Beacon was awarded as the most excellent performer among several other controllers (NOX, POX, Maestro, Floodlight, Ryu) in the study.

The issue of software aging was highlighted by the research conducted in [62]. The main issue studied was memory leakage between two Java-based controllers (Floodlight and Beacon). Floodlight was beaten by Beacon since the latter displayed less memory consumption. In this study, the most common open-source controllers were compared based on multiple criteria. Thus, the requirements laid out by the researcher will influence the result in selecting the best controller.

Authors in [64] shared their experiences and lessons learned from building two ONOS prototypes based on performance, scalability, and availability. The idea is to verify the core features and improve the performance of remote data operations. Research revealed how ONOS as a distributed SDN controller able to meet the criteria of production networks. Two use-cases of simple proactive route maintenance, and BGP interfacing application deployed over ONOS platform. The authors also explored other use cases in packet optical core networks (traffic engineering and scheduling), next-generation points of presence (PoPs) focusing on virtualization, network resources (computers, storages and networks), and customer-based networks. ONOS also provides an abstraction of connectivity that leads to the ability to handle the mechanism of network topology, host location, or change of utilization. This is based on their experience deploying SDN-IP peering applications.

The authors in [65] conducted a performance evaluation of the POX controller and OpenFlow using mininet simulator using service delay, utilized bandwidth, received packets and bytes as metrics and iperf and D-ITG as network monitoring tools. The authors recommend using the POX controller for rapid development and prototyping of a network control system and its use as the framework for interaction with open-flow switches.

Using mininet, the authors in [66] evaluated the performance of pox and RYU SDN controllers with D-ITG used for performance evaluation and iperf for measuring maximum available bandwidth. Results show the RYU controller yielded better performance with respect to coverage, delay, jitter, bit rate, and throughput, and the application requirement should dictate the selection of the controller. The authors in [67] conducted a performance analysis of a simple IDS implemented in an SDN environment, analyzing the resulting CPU usage and memory allocation of the controller. The authors in [68] studied the performance analysis of a load-balancing algorithm using the POX controller. They set up experimental tests to implement and compare four load-balancing algorithms (Random, RR, WRR, LC). Using mininet emulator and OpenFlow switch which was connected to POX. This is evaluated based on the degree of load balancing, workload, and time values by generating different numbers and sizes of packets. The results show the accuracy of the workload metric and average RT metric.

The authors in [69] studied the performance analysis of floodlight and RYU under mininet simulator. Using tools like mininet and qperf. The bandwidth and delay of floodlight and Ryu were compared under different network topologies. Floodlight was shown to have higher bandwidth and

lower latency than the Ryu controller. The authors in [70] studied the implementation and performance analysis of the SDN firewall on the POX controller. The authors used POX and Open vSwitch while virtualBox and mininet were used to create SDN topology while wireshark and iperf were used to analyze the performance of the firewall. It implements some firewall (which works at layer 2-layer 4 which can detect traffic and enforce rules) function on SDN via writing a firewall application that runs on top of the POX controller.

The authors in [71] implemented SDN architecture using an open-source RYU SDN controller for analysing the network traffic. The performance of the SDN architecture was evaluated using a custom network topology for node-to-node performance parameters, e.g., bandwidth, throughput, and RTT. The proposed work performs better than the default SDN network topology. The authors in [72] studied the performance analysis of two OpenFlow-enabled controllers (floodlight and ONOS controller) over both linear and tree topologies using metrics such as transfer, delay, bandwidth, and jitter using mininet. The results showed that ONOS performed better compared to floodlight in TCP and UDP traffic. The authors in [73] conducted the performance evaluation of POX and floodlight using mininet. Floodlight was shown to outperform POX in terms of throughput. The authors in [74] study the use of an RYU SDN controller-based test bed for testing the performance of the source address validation technique. Data packets are forwarded using the destination address without validation of the host address which makes SDN vulnerable. The proposed testbed worked well based on the sequence of source address validation techniques.

The authors in [75] conducted a performance comparison of RYU and floodlight controllers using mininet network emulator. The authors in [76] conducted a performance comparison of RYU and floodlight controllers under different SDN topologies e.g., single, linear, tree, torus and custom for throughput, jitter, latency, and packet loss. RYU was shown to yield better throughput compared to floodlight in all topologies. Except for Torus, RYU performs better with respect to latency and jitter.

The authors in [77] conducted a performance analysis of RYU and POX controllers with latency and throughput using simple tree-based and fat tree-based network topologies. Using mininet, the authors develop an SDN model. The authors show that RYU yields a better performance than POX and thus it is suitable for small-scale SDN while for throughput, POX performed better showing it can respond properly under FTB traffic load but with greater utilization of hardware resources. POX can respond to requests faster under complex FTB traffic loads, at the expense of higher hardware resource utilization.

The authors in [78] conducted performance analysis of POX vs floodlight over different network topologies using throughput, RTT, and delay using mininet single, linear, tree end user-defined topologies. Floodlight performed better with respect to POX using RTT as well as throughput as metrics. The authors in [79] performed an extensive performance analysis of ODL and ONOS using mininet then wireshark packet analyzer and iperf were used. Particularly, iperf provides real-time traffic flow between mininet and controller. ONOS was shown to perform better wrt throughput, TCP under scaling, TCP/UDP bandwidth burst rate, jitter, goodput, RTT, usability and TCP stevens graph.

The authors in [80] conducted a performance comparison of two popular controllers, floodlight and Opendaylight, in terms of delay and loss under different topologies and network load under different use cases. OpenDL was shown to outperform floodlight in low-loaded networks and for tree topologies in networks with medium load for latency while floodlight performs better in networks with heavy load for tree topologies (wrt packet loss) and linear topologies (wrt latency). No significant differences were observed in other cases. The authors in [81] studied node-to-node performance

evaluation and test analysis with respect to throughput, RTT, etc. via RYU SDN controller. This is achieved using mininet containing a RYU controller with a switching hub, one OpenFlow switch and 3 nodes.

The authors in [82] studied the performance of floodlight and POX under different scenarios. This was conducted by examining mobile wireless station connectivity with the fixed wired station using the mininet Wi-Fi emulator. The authors in [83] conduct a performance evaluation of distributed SDN using floodlight controllers. The paper sheds light on SN challenges, including the ability to scale and to be fault-tolerant. The authors in [84] study the effect of different link failure scenarios on DC network performance using Ripl-POX controller using a 4k-fat tree DC network on a mininet environment.

Another important aspect of performance evaluation is the use of multi-criteria decision-making methods. These approaches have led to interesting results and comparisons for different controllers [85–91].

Controller selection is one of the most important issues in SDN networks. The authors in [85] divide the features of SDN controllers into two categories. Thus, regarding how diverse SDN controllers are, the authors consider answering questions relating to how appropriate SDN controllers should be chosen. This question is challenging because of the diverse characteristics of SDN controllers, making it more difficult to arrive at the most accurate decision. For this reason, the authors deploy multi-criteria decision-making. The authors compare POX, NOX, Beacon, Floodlight, Ryu, ODL, and ONOS. Deploying MCDM, particularly the best-worst multi-criteria, they find the most appropriate SDN controller. They solve an optimization problem and study the performance with throughput and latency, and the result of the initial evaluation revealed that ONOS and ODL yield the highest throughput while NOX, POX, and Ryu yield the lowest throughput. The final evaluation using all criteria confirmed that ONOS and ODL were robust compared to open controllers.

Considering that every controller supports several features, some features may be more prominent in a particular controller. Then, the authors in [86] leveraged the analytical network process to rank the SDN controllers based on their features. Thereafter, they created a hierarchical control plane cluster of the top two controllers that yielded the highest weight. Internet OS3E topology was used in the experimental evaluation. The results show that ODL which has a high weight, outperforms ONOS with a lesser weight when the controller is used without clustering and when the proposed HCPC approach is applied. HCPC with ODL performs better than ONOS and DCC for delay, CPU utilization, jitter, load balancing, recovery time, and scalability.

The authors in [87] study the specifications of most of the open-source NOS and categorize the features into two groups: non-functional features and functional features. Non-functional features include security, interoperability, maturity, ease of use, scalability, and availability. Functional features include traffic protection solutions, packet forwarding techniques, troubleshooting and fault verification, and virtualization. They used AHP, a decision support system, to assess the specifications of ONOS, Ryu, Floodlight, POX, and Tungsten. The objective is to find the best NOS for CDC via an assessment of the specification of ONOS according to the criteria of the requirements of the cloud data center. Results show that ODL is the best NOS for cloud data centers. Also, ODL and ONOS yield similar scores when compared to the other network operating systems.

The authors in [91] deploy entropy-based TOPSIS to select the best controller for load balancing in the control plane. Four controllers (POX, Beacon, Floodlight, and RYU) are selected and evaluated over tree topology considering throughput, delay, response time, and message cost via mininet simulator. The results using TOPSIS to rank controllers show that Floodlight yielded the best

performance when compared to other controllers in a scenario involving 5000 packets analyzed under a tree topology.

6.2 Monitoring

The research done by [65,92] addressed monitoring in the ONOS framework, which has a multi-controllers feature. Real-time monitoring results were provided with the recording of OpenFlow messages exchanged on the ONOS logging system by the former. The overall load of individual controllers working in a logically centralized SDN environment was dynamically monitored by an adaptive monitoring solution by the latter. However, neither monitoring wireless-specific metrics, nor network performance metrics were given any attention.

The introduction of SDN technology has catalyzed the development of many SDN controllers. The initial design of a single SDN controller has problems in managing large-scale networks due to high traffic congestion in the control plane. In order to address this issue, the Open Network Operating System (ONOS), which is designed based on the concept of a distributed SDN controller was then proposed. However, due to a lack of performance assessment to determine how the workload is to be distributed, ONOS not able to solve the congestion problem. Studies have been done on how to monitor ONOS performance. Works from [92,93] focused on OFMon, the first monitoring system in ONOS to detect and monitor activities of OpenFlow messages on ONOS controllers and OpenFlow-based switches. Further to this study, an experiment was performed to test and evaluate the CPU and memory usage of OFMon. Performance results showed very little network resource utilization, with a maximum of 4% more CPU and 6% of memory usage recorded by OFMon.

The authors in [94] study the use of floodflight controllers as SDN traffic monitoring in data center networks. The tool does not require additional traffic as it depends on traffic querying. The results show the successful detection of elephant and cheetah flow which can be re-routed to improve QoS.

6.3 Topology and Architecture

Authors in [23,95] suggested that by decoupling the physical network in terms of topology, address and control functions, multiple tenants can share the same physical infrastructure and create independent virtual networks (VNs). The programmability feature of the SDN paradigm may pave the way to materialize full network virtualization (NV). Many advantages have been brought about by SDN to both network operations and management such as programmability, agility, elasticity, and flexibility. Nevertheless, existing SDN-based NV solutions still have some drawbacks in terms of scalability and high availability. The proxy-based architecture also introduced high latency between the planes and thereby added a list to the existing problems.

In their thesis [95], they introduced a new NV platform, named Open Network Hypervisor (ONVisor) as a step towards a scalable and flexible SDN-based network virtualization by extending ONOS. Among the features in the design objectives are (i) multi-tenancy, (ii) scalability, (iii) flexibility, (iv) isolated VNs, and (v) VN federation. As research in solving existing NV platform issues, ONVisor sets out some of the key criteria of (i) remote control and data plan per VN, (ii) distributed operation support, (iii) extensible translators, (iv) on-platform development and implementation of VN applications and execution, and (v) support of diverse SDN data-plane implementations. Experiments were carried out to compare the NV platform to a non-virtualized SDN network in terms of control and plane performance in various test scenarios and environments. Outcomes from the experiments showed that ONVisor can provide only a slightly lesser control plane performance and similar data

plane performance to VNs. The authors in [96] proposed an SDN architecture for IoT network that is based on an ONOS controller using mininet.

6.4 Security

In [97], Sandra Scott-Hayward discussed the evolution of SDN security. Although SDN technology has progressed rapidly in the past few years, the emphasis on security was not moving at the same pace. The researcher focused on the security evolution of the two most popular SDN controllers; ONOS and ODL, which have a broad deployment and strong contributor base. Although the focus on security has increased within both controller communities through security support and new features introduced with each new software released, the lack of security integration should raise concern among the developer community. Although there is a demand for more secure, robust and resilient controllers especially for public networks, the response to a more secure controller design is still very limited.

The authors in [98] analyzed the detection and mitigation of DDOS attacks with RYU controller. They study the early detection and mitigation of DDOS on services using ML models. This was used to minimize the prediction time and ensure the correctness of the dataset and model accuracy. The authors in [99] performed a security analysis of ODL and ONOS controllers and decreased vulnerability issues in SDN controllers. The authors attempt to address the security vulnerability of SDN with ODL and ONOS controllers for DDOS attacks.

The authors in [100] focus on the implementation of SDN traffic monitoring using the RYU controller. RYU was used to obtain the flow of information of the switch via OpenFlow protocol to obtain the load status and remaining bandwidth of the network link. The result is useful for SDN congestion control and load balancing. The authors in [101] performed a security analysis deploying packet sniffing and spoofing on POX and RYU controllers. Layer 2 security test was conducted on POX and layer 3 on RYU where packets are filtered based on packet type. Results show that RYU is among the most comprehensive programmable controllers for providing security.

The authors in [102] perform a security analysis of Opendaylight ONOS, Rosemary and RYU SDN controllers, OpenDaylight was shown to be the most secure. The authors provide a summary of current security developments concerning SDN. The authors in [103] provide a comprehensive resilience analysis of ONOS recovery and openDL recovery time when there are link and switch failures. The authors in [104] study the security of floodlight zerosSDN, beacon and POX using the STRIDE threat modelling technique. SE floodlight was shown to be the most resilient. The authors in [105] summarize SDN security issues for future SDN mobile networks. Experiments were also presented for network topologies using network attack scenarios to showcase how security could be taught using SDN controllers. ONOS was used to perform experiments showing security can be easily taught using ONOS.

7 Discussion

In this section, we aim to answer the following questions relating to the implications of the open-source nature of SDN controllers with regard to vendor lock-in, interoperability, and standards compliance, real-world use cases and success stories of organizations implementing open-source SDN controllers in their networks, how the open- source community contribute to the development and improvement of SDN controllers, and the key challenges faced by open-source SDN projects. Similarly, considerations organizations should take into account when choosing an open-source SDN controller for their specific networking requirements and goals, and how emerging technologies

like Artificial Intelligence (AI) and Machine Learning (ML) impact the evolution and capabilities of open-source SDN controllers. In addition, the challenges and limitations associated with deploying open-source SDN controllers in production networks, how can they be mitigated, and finally how open-source SDN controllers handle network security and ensure that network configurations and policies are robust and resilient. Also, we would discuss some other challenges and future considerations.

7.1 Implications of Open-Source SDN Controllers

Although SDN brings benefits concerning manageability and the automation of network processes, amongst others, vendor lock-in, investment in SDN-capable hardware, and backward incompatibility are some of the major challenges [106]. Particularly, a non-open, i.e., closed or proprietary technology, leads to vendor lock-in problems where users end up with fewer or no options but to use the legacy proprietary controllers or tools. Particularly with the advancement of technology, vendors enhance, upgrade, and develop equipment. However, this leads to issues in interoperability between different devices, which "compels" the customer to use the equipment of the present vendor or change all devices, which is at a huge cost. An example is that if two nodes in a network may not see each other (be visible) on the same management GUI, they may need more than one application to handle them [107]. Usually, the same vendor provides devices and network management tools. This leads to issues that make vendor lock-in problems deepen as the network evolves. However, with open technologies, there is higher network flexibility, extended choices, innovation, and a reduction in vendor lock-in since open software can control multi-vendor assembly of different components through open and standard interfaces [108]. Therefore, companies can re-use open source applications or vendor-agnostic applications in the application plane, and there is a possibility to write code in different languages for performing intelligence, optimization, and new services [109] leading to innovation. As such, when many vendors use a particular open SDN controller, it is viable and easier to achieve cross-vendor interoperability [110]. Finally, APIs can be used to facilitate network automation and micro-service function orchestration to cater to the requirements of different applications. This can facilitate interoperability between network application developers of the different sub-functions of the SDN controller [111].

As regards standardization, it is important to comply with standards and resolve software issues against a service level agreement. Large telecommunications companies need service guarantees to softwarize their network (using open controllers). Such Softwarization helps to deliver new services as it makes programming and deployment more flexible; however, standardization, which provides stability, more interoperability, and flexibility and triggers more innovation, comes with some trade-offs. Softwarization can facilitate rapid innovation, whereas standardization moves solutions towards more harmony and is less flexible concerning the outcomes [112]. However, when experts contribute actively to open controllers, the need for interoperability could foster faster standardization while accommodating variety via community participation, review, and discussions. However, this requires a lot of organization and coordination within the open-source community.

7.2 Real-World Use Cases and Success Stories

One of the main purposes of SDN is its availability to all as open source without hidden proprietary components. This helps to facilitate the independent use and growth of SDN. Some hardware companies are now forced to consider open-source options as some of their competitors have found a place for open standards in their production lines [113].

7.2.1 Examples of Use Cases Using OpenDaylight

A few real-world use cases of open-SDN using OpenDaylight involve different telcos, academic and research institutions, and enterprises. For instance, for automated service delivery, these include AT&T, Caltech, globo.com (accessed on 10/01/2024), KT, and Orange. For cloud and NFV, companies include Cable Labs, CenturyLink, China mobile, globo.com and Orange[1]:

- Telstra's PEN platform empowers users to build on-demand high-performance networks with the required scalability and flexibility for building hybrid clouds. It is cost-effective and reliable, and PEN pioneers the way with SDN all around the globe to enable businesses to self-provision dynamic network services.
- The high energy physics researchers at California Institute of Technology (Caltech) are among the large network of researchers around the globe who are performing experiments using the Large Hadron Collider, the biggest machine in the world at CERN and France for making new findings about the evolution of the Universe using Linux and open source software.
- Globo.com is a company based in Brazil, Grupo Globo which provides internet-related services and platforms to the companies in the group. Globo.com's aggressive innovation was motivated by the network scale needed to support the company's internal cloud. As the main controller, OpenDayLight was chosen due to its large open-source community, good documentation and well-defined interfaces for both users and services.
- KT Corporation is the largest telecommunications service provider in South Korea and offers high-speed internet access, wireless services, and wireline telephony. It was previously known as Korea Telecom and was founded in 1981. It deploys the OpenDaylight SDN platform for creating and deploying new, flexible and scalable Transport SDN WAN network as the leading nationwide carrier WAN services.
- Orange has been the leading supporter of open standards with a long history of participation and contribution to open-source communities. It is one of the largest telecommunications operators around the globe, providing a wide range of both mobile and fixed services to more than 247 million users in 29 countries. Orange has been actively participating in the community since OpenDayLight was launched in 2013. Orange is a founding member of ONAP, OPNFV and a platinum member of LFN.
- A research and development consortium that is not profit-based, CableLabs is dedicated to the creation of novel ideas that impact the business of its international member cable operators. They are prototyping SDN and NFV use cases to define interoperable solutions among their members and technology supplies. This is aimed at reducing costs, creating competition in the supply chain and driving scale.

CenturyLink's full commitment was to virtualize its IP core network by 2019 and they are emulating the work in some open-source communities to create a central office re-architected as data center.

7.2.2 Some Other Projects Using Open-Source SDN Controllers

The first milestone of ONF was recently achieved in the SMaRT-5G project. Particularly, this is with the demonstration of an intelligent cell switching (ON/OFF) radio access network energy savings application that works together with the xApp traffic steering application to ensure QoS while optimizing energy consumption. These applications are implemented entirely in open source. Both

[1] https://www.opendaylight.org/use-cases/stories (accessed on 10/01/2024).

(energy saving and traffic steering) are two important mobile industry use cases which are combined as a unified open source solution for the first time by the ONF community [114].

Big Switch has an open programmable SDN product suite which makes it easy to adapt new network applications as compared to traditional non-programmable networks. Open standards and APIs are supported by the hardware platform-independent suite by Big Switch. Big Switch uses OpenFlow and Floodlight, which are two popular open sources for providing abstraction for the physical infrastructure, policy-based functions, and central intelligence for SDN/programmable-based networks. HP also offers OpenFlow-enabled controllers and switches, indicating its support for open standards [113].

Broadcom has SDN technologies with open-source initiatives from major industry experts aiming to make networks more scalable, flexible, and programmable and improve infrastructure performance. Its SDN technologies support several network management procedures and applications while also improving optimization, harnessing network-level control, and reducing network complexity. Broadcom worked closely with ONF on developing the OpenFlow protocol and has played a pivotal role in the development and demonstration of OpenFlow running at scale [115].

China Mobile pioneered packet-based transport networking and is leading the introduction of SDN for future-generation packet-based transport networks. They are collaborating to unleash the full potential of OpenFlow over merchant silicon and working to develop technologies for transport networking [115].

According to the CTO at Deutschland Telekom, Walter Goldenits, open-source projects are at the forefront of the promising movement enabling their industry to manage data growth efficiently [116]. Thus, Deutsche Telekom, one of the leading integrated telecommunication networks in the world, with about 165 million mobile customers, 18.5 million broadband lines, and 28.5 million fixed-network lines, joins AT&T, China Unicom, Google, NTT Communications, Comcast, and Verizon as one of the leading operators that guide ONF's mission to drive innovation in operator networks while concurrently working towards the transformation of business models across the industry. DT began a live trial in Berlin, putting the SD-RAN open-RAN project by ONF to the test. The trial features disaggregated hardware, including central units paired with open-source nRT-RIC software and xApps developed as part of the SD-RAN project. Deutsche Telekom is one of the companies that joined the ONF as a full partner member, which is the highest tier of support and investment in ONF and a significant indication of an operator's help to drive the mission and impact of ONF. ONF has over two hundred members, which gives it a large breadth of solutions. The open initiative makes it much easier for vendors to identify operator use cases, contribute to solutions, and take the solutions to operator trials. Vendors can identify new markets and take advantage of opportunities by leveraging the broader communities' work to reduce the cost of research and development while accelerating the time to market using the solutions [116].

Cisco and some of its partners addressed many challenges in the UK 5G Rural first project by deploying 5G at a lower cost than ever using the Cisco open software-defined architecture [117]. The Agile SDN Controller of Huawei is based on ONOS and they also ensured it was compatible with ODL using API [2].

[2] https://www.sdxcentral.com/ (accessed on 10/01/2024).

7.3 Contribution of Open Source Community and Challenges

The open-source community contributes differently to the growth and development of open-source controllers. These could be in terms of skills, financial support, and ideas. Codes are edited and reviewed by a wide variety of contributors; documentation is provided via community collaboration; large players in the industry are participants; discussions are made and suggestions are provided; documentation and code are reviewed; new and updated versions can be released with contributions from the community; bugs could be fixed within the larger community; a wide range of support can be provided; support for different languages [118] can also be accommodated; and many more. Different experts all around the globe can contribute to these open-source projects; mentoring and supervision of new entry-level technical support teams are also provided. The result is controllers that achieve flexibility, progress of different projects towards standardization, adaptability to different use cases and needs, and a wide range of support. Similarly, for instance, in some cases, the ODL open-source solution [119] has been able to support several global network subscribers. In summary, the open-source community helps to develop solutions that are agile at a fraction of the cost of traditional proprietary solutions [120].

Challenges regarding the open-source SDN community include coordination of input from contributors and deadlines; financial support could be inconsistent; participation can sometimes be skewed toward specific parts of projects; and new hands might not be forthcoming on other parts. A review of documentation could take a significant amount of time; delays could be experienced in fixing bugs and releasing new versions; and uncertainties might be experienced in the time needed to get community support. Also, a significant amount of time may be required to validate different architectures and interfaces to ensure interoperability and consideration for mass market deployment by companies. Other challenges experienced include, in some cases, inconsistent maintenance of the open SDN project, need for constant and continuous tracking of dependencies, ensuring software maturity, ensuring constant and consistent (verification and) approval of changes, and ensuring dependencies are constantly looked into. Also, ensuring the reliability and security of the open source projects and constantly reviewing compliance, continuous review of code for vulnerabilities, slow patch development, and compromise of code packages [121] are all inherited from challenges in general open source projects.

7.4 Considerations for Choosing Open Source SDN Controllers

Choosing the right open-source technology is a critical decision and plays a major role in improving scalability, efficiency, and the overall success of the project [122]. There are so many factors to consider when choosing an open-source SDN and open-source solutions in general: code quality or maintainability, security risks (as well as breaches), and vulnerability. Does vulnerability have patches or solutions [123]? Similarly, several requirements can be extended to SDNs too [122]. Project requirements and objectives should align with those of the open-source controllers; it is also important that the open SDN controller can handle the projected growth of the company's demands (to avoid bottlenecks). In addition, is it easier to deploy, and does it provide the best experience? Is there active and continuous community support? Is there technical expertise and familiarity with the frameworks and languages of the controller? Is there a prospect for longevity of relevance and adaptation to future trends and expectations? Similarly, market response, reliability mechanisms, interoperability with current solutions, the existence of skilled talents to optimally deploy the controllers, larger and richer discussion and support forums, and comparison with paid solutions (for comparisons, especially when paid expertise is required to efficiently deploy open source solutions), then who takes responsibility for risks associated with the controller? The issue of security is broad, and it is most important to

understand how open-source solutions can match the security priorities of the organization [124]. Other issues include the diversity of the community involved and the level of expansion of the project, and terms and conditions of use [125].

7.5 Role of AI on the Evolution of SDN Controllers

AI plays a vital role in different aspects of open SDN controllers and their development, particularly in improving the functionality of SDN controllers. Such functionality includes network monitoring and security, load balancing, network virtualization and orchestration, and policy enforcement. These four aspects [126] are vital. For instance, network monitoring is one of the most important use cases of SDN since the controller leverages the bird-eye view of the network topology and can thus query the network performance more proactively. With AI, this becomes even more efficient due to the prediction capabilities of different AI algorithms [126]. Also, load balancing plays a very crucial role in optimizing different performance aspects and measures, including the minimization of response time, maximization of throughput, and optimization of resource allocation [126]. SDN controllers help to facilitate efficient network virtualization by enabling network slicing and multi-tenant hosting on existing physical resources. Thus, AI-aware optimization techniques can make this more efficient. Policy enforcement is another important functionality of SDN controllers, as efficient rules can be implemented with the assistance of AI techniques that can learn the system to suggest efficient policies to be applied for optimal results [126]. Thus, AI is promising to largely influence the future of the development of SDN controllers. Currently, one of the most popular use of AI is security for DOS Attack detection and mitigation, as done in [127–143], then anomaly detection in general [144,145], network management and intrusion detection [146–149].

Other scenarios where AI has been featured include: achieving scalability [150,151], load balancing [152], optimization [153], prediction-based controller placement [154], traffic classification and prediction [155–157], performance trade-offs of flow monitoring in SDN [158] load balancing, routing and policy optimization [159], resource allocation [160,161].

7.6 Challenges Associated with Deploying Open-Source SDN

Deploying open-source SDN in production networks can be quite challenging. Even before deployment, one of the most prominent challenges is identifying which open-source SDN controller is most suitable for production, considering its strengths and its compatibility with the production application requirements and objectives, and whether consideration should still be given to using other compatible proprietary tools and solutions in addition to the open-source controllers. Determining the scalability of the open-source controllers to accommodate growth and network expansion is another key challenge. Ensuring the deployment is hack-free and not vulnerable to security threats. Ensuring ease of adoption of the open source controller as well as the maturity of the controller over time by significant community contributions. Another is ensuring that the community is responsive to queries and responds quickly to issues discovered by others. Determining the adaptability of the open-source solutions and the convenience with which they could be customized or extended to address the peculiar needs of the company. Also, determining the potential risk of malicious code introduced by third parties in the code. Some others include navigating customization, deployment, etc., determining whether it meets current industry standards, and how interoperable the controllers are with the existing solutions in the production chain. Finally, another challenge relates to handling network security, especially due to the security risk that may be incurred or inherent, and ensuring that network configurations and policies are robust and resilient.

Ensuring that network configurations and policies are robust and resilient can be achieved through proper traffic monitoring, community involvement, policy optimization and enforcement, and integrating security frameworks. AI can play a vital role in determining optimal policies and configurations for the best results, as well as prove useful in anomaly detection and attack mitigation.

7.7 Other Challenges and Future Work

The evolution in the network architecture of SDN will create new potential and challenges. Several considerations should be taken to determine the best-suited solution for the implementation of an SDN solution. Listed below are several potential challenges and opportunities:

- Standardization is one of the key issues around SDN. Standards bodies like the Open Networking Foundation (ONF), the Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T), the Internet Engineering Task Force (IETF), and the Optical Internetworking Forum (OIF) are involved in different areas of SDN. Unlike some network technologies, such as Ethernet, there is a need for standard bodies responsible for coordinating open standards for SDN. To a slight degree, there are strong initiatives from industry players in developing standards and guidelines for SDN. Open standards can easily provide a platform by which some of the benefits of open-source SDN controllers can be fully harnessed such as interoperability, and foster innovation by harmonizing inputs towards landmark improvements that could set the pace even ahead of proprietary SDN controllers.
- Centralization of the controller creates a single point of failure, scalability issues, and a lack of efficiency. The scalability and availability of using a centralized SDN controller may lead to security issues. As the intelligence of SDN logically centralizes in the controller, the network is vulnerable to malicious attacks and system reliability. Earlier development of controllers is more focused on research and small-scale operation which does not focus more on high availability and reliability. As the requirement expands to the service provider environment, the design of a controller such as ONOS is built for carrier-grade specific as such clustering support. Thus ensuring a very robust management and proper security protection of the controller is of paramount importance and the developments in machine learning and artificial intelligence can play a huge role in the prevention, detection, and mitigation of security attacks (such as DDoS) at the controller. Similarly, the optimization of different network performance metrics related to the centralized control in the network is another consideration that ought to be further given attention.
- Regarding Application Programming Interfaces or APIs, at the moment no dominant protocol standards apply. Most SDN controllers inclusive ONOS support OpenFlow as the standard southbound interface of SDN architecture but there are still challenges with the industry acceptance. While most of the SDN controllers are OpenFlow ready, they are also built with legacy southbound interfaces including existing BGP, Netconf, XMPP, OVSDB, MPLS-TP and many more. OpenFlow may not remain the choice of SDN protocol. SDN controller APIs dictate how the application is written and exposed and thus more variety with regard to standardized APIs would be required bearing in mind the importance of interoperability in open-source solutions.

SDN industry is seen to be growing in tandem with the diversity of initiatives and projects that have been brought together despite the challenges posed above.

- While standards have been challenging, a significant effort on open source initiatives and industries supports the future SDN standard emerging trends from several standards bodies.

ONF, for example, is very aggressive in collaborating with standards bodies and technology vendors to come out with common standards that could lead to single standard references for the SDN industry. A lot of standards initiatives activities towards SDN and many of those standards are still emerging.

- Since the current providers' network is composed of multi-layer technology with different vendor solutions and control protocols, SDN eliminated the complexity by creating centralized control and directly programmable regardless of the underlying network infrastructure of the network. Thus, gain the ability to reroute data traffic on the fly, and enhance network performance and agility over different network applications (Bandwidth on Demand, Load balancer, Deep Packet Inspection, DDoS mitigation, etc.) and services.
- SDN has the potential to be used in diverse fields and sectors. Apart from telecommunications and networking, aviation traffic has begun to apply the concept of SDN. The security industry of Air Traffic Safety (ATM) has practiced the concept of ONOS SDN, driven by Air Navigation Service Providers (ANSP), Frequentis' NetBroker built on ONOS in broadband optimization and automated routing. In 2017, ONOS-based NetBroker was being rolled out in Brazil [52].
- SDN is well suited to operate in the cloud and virtualization environment. SDN and Network Function Virtualization (NFV) both aim to simplify while automating the traditional way that network operators build and manage networks. For SDN to work well in the virtualization layer, it needs to go hand in hand with NFV. If SDN is about logically centralized network control, extracting complicated parts of the typical network environment, NFV is about network virtualization, the orchestration of network functions, and management. SDN and NFV are highly complementary [52].
- ONOS is the latest open SDN controller that focuses on service providers' needs and the only SDN controller that focuses on carrier-grade compliance, completely open source, fit high-performance aspects, and has the reliability to increase the availability and scalability of service provider network critical requirements and compliance with open standards. Respecting the openness and scalability criteria, ONOS has quickly matured and been adopted by carrier-grade networks and telcos.

The future of SDN is open source and also the future of network providers. Looking to the future, provider networks are no more vendor lock-in-based. Companies are aggressively involved in open-source projects and run their business operation through open-source solutions as the underlying technology. SDN is introduced as the solution to today's network complexity. Decoupling network control into separate centralized devices and the ability of the operation to be operated through virtualization and standard protocols may result in less capital and operating costs. Furthermore, ease of management from a centralization controller, and become directly programmable of the devices. Open standards become so important for companies to move further and be more innovative. There is a strong transition of legacy to a neutral network. One of the latest SDN open-source projects is ONOS, which is an SDN controller deployed by the industry.

The future of SDN is believed to diversify existing technology variations to become more prominent in improving programmability and automation. Thus, many options are offered in SDN variants such as SDN-WAN and SDN-DC. These variations exist due to several factors such as (i) The provision of many public and private clouds; (ii) Improving automation to reduce mean time to deliver and restoration; (iii) The need to monitor and operate a hybrid infrastructure over physical or virtual network; (iv) Product change and service monolithic to micro-service; (v) High demand using white box infrastructure and lower-cost options.

8 Conclusion

The software-defined networking offers affordable opportunities for service providers to build and adopt various types of SDN solutions. Growing operators and vendors have begun to take more resource adoption of the open source domain as the trends move towards deployment plans using SDN technologies as a new way of designing, controlling, and managing networks. The main target is to have a more scalable, agile, and reliable networking system with a software-based approach. One of the key factors for growing the support of open-source SDN controllers is various organizations' commitment to the use of open-source or build controller platforms based on open-source solutions. This would create a complete and open networking ecosystem. Thus, operators and vendors should take advantage of the open-source SDN controller platform.

The open-source SDN controllers, ONOS provide an alternative to service providers to deliver carrier-grade networks. ONOS has progressed tremendously and has been accepted by carrier-grade networks and telecommunication companies due to their openness and scalability criteria apart from the criteria discussed in the previous section. ONOS was completely open source, fit high-performance aspects, and the reliability to increase the availability and scalability of service provider network critical requirements and compliance with open standards. ONOS focuses on service providers' needs and is the only SDN controller that focuses on carrier-grade compliance. ONOS is designed and developed for scalability, high performance, and high availability.

This paper compares five popular and commonly deployed open-source controllers which are NOX/POX, Ryu, Floodlight, OpenDayLight, and ONOS. Each of the controllers is briefly introduced and compared in terms of the origin, programming language, carrier-grade support, learning curve, etc. ONOS has been created with (i) carrier-grade features such as scalability, high availability and performance in terms of throughput (application intents per second) and latency (time to process network events); (ii) northbound abstraction/APIs to ease the creation of new services using ONOS by extending the agility of software to networks; and (iii) southbound abstraction with device and protocol plugins as a means for ONOS to provide SDN control over OpenFlow enabled devices and legacy devices. It allows a smooth transition to an SDN-based network operating over white boxes or any community servers. This paper explores and provides relevant information that ONOS has filled a critical role in the service provider network by providing a set of high-level abstractions and models, which it exposes to the network elements and applications layer while following emerging open standards. Leading service providers and vendors have also assisted ONOS by guiding it with use cases. The network industry giant and leading service providers have planned and deployed SDN and ONOS in their lab environments and brownfield operation networks. Thus, ONOS will be the stronger choice for service providers that need to meet carrier-grade networks yet require open SDN solutions to the industry.

This paper has also provided answers to several important questions relating to the implications of open-source SDN controllers on standard compliance, interoperability, and vendor lock-ins as well as examples of the use of open-source SDN in real projects, the contributions of the open SDN community, the role of AI, and how security issues are handled. Particularly, there have been a lot of studies on the use of AI to detect distributed denial of service attacks. This paper has also provided some of the recent studies on open-source SDN controllers concerning performance evaluation, traffic monitoring, topology and architecture as well as security. Challenges associated with open-source SDN and future directions have also been provided.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: Sunway University funded the APC for this paper. Oluwatosin Ahmed Amodu's research is supported by Universiti Kebangsaan Malaysia, under Dana Impak Perdana 2.0. (Ref: DIP–2022–020).

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Johari Abdul Rahim, Rosdiadee Nordin; data collection: Johari Abdul Rahim, Oluwatosin Ahmed Amodu; draft manuscript preparation: Johari Abdul Rahim, Oluwatosin Ahmed Amodu; Supervision: Rosdiadee Nordin; Funding: Rosdiadee Nordin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and codes that support the findings of this study are available from the corresponding authors upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. Duck and N. Bridge, "Future of open source survey," 2016. Accessed: Apr. 25, 2018. [Online]. Available: https://opensource.com/business/16/5/2016-future-open-source-survey

[2] S. Krishnamurthy, "An analysis of open source business models," in *Perspectives on Free and Open Source Software*, pp. 279–296, 2005. doi: 10.7551/mitpress/5326.003.0022.

[3] M. Mustonen, "Copyleft—the economics of Linux and other open source software," *Inf. Econ. Policy*, vol. 15, no. 1, pp. 99–121, Mar. 2003. doi: 10.1016/S0167-6245(02)00090-2.

[4] M. Godfrey and M. Tu, "Growth, evolution, and structural change in open source software," in *Proc. 4th Int. Workshop on Princ. Softw. Evol.*, ACM Press, 2001, pp. 103–106.

[5] S. Koch, "Evolution of open source software systems–A large-scale investigation," in *Proc. 1st Int. Conf. Open Source Syst.*, Genoa, Italy, Jul. 2005, pp. 148–153.

[6] G. Robles, J. J. Amor, J. M. Gonzalez-Barahona, and I. Herraiz, "Evolution and growth in large libre software projects," in *Proc. Eighth Int. Workshop on Princ. Softw. Evol. (IWPSE 2005)*, IEEE Computer Society, 2005, pp. 165–174.

[7] C. K. Roy and J. R. Cordy, "Evaluating the evolution of small scale open source software systems," *Advances in Computer Science and Engineering*, vol. 123, 2006. Accessed: Apr. 25, 2018. [Online]. Available: https://research.cs.queensu.ca/home/cordy/Papers/RCS_RoyCordyEvolution.pdf

[8] G. Succi, J. Paulson, and A. Eberlein, "Preliminary results from an empirical study on the growth of open source and commercial software products," in *EDSER-3 Workshop*, 2001, pp. 14–15.

[9] Grand View Research, "Software defined networking market size, share & trends analysis report by end-use, by service, by solution, by application, by region, and segment forecasts, 2018–2024," *Market Analysis Report*, Report ID: 978-1-68038-114-6.

[10] C. Prabha, A. Goel, and J. Singh, "A survey on sdn controller evolution: A brief review," in *2022 7th Int. Conf. Commun. Elect. Syst. (ICCES)*, Coimbatore, India, IEEE, 2022, pp. 569–575.

[11] S. H. Darekar, M. Z. Shaikh, and H. B. Kondke, "Performance evaluation of various open flow SDN controllers by addressing scalability metric based on multifarious topology design on software-defined networks: A comprehensive survey," in *Proc. Third Int. Conf. Intell. Comput., Inform. Control Syst.: ICICCS 2021*, Springer Nature Singapore, Mar. 2022, pp. 327–338.

[12] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers," *J. Netw. Syst. Manag.*, vol. 29, pp. 1–59, 2022.

[13] K. Nisar *et al.*, "A survey on the architecture, application, and security of software defined networking: Challenges and open issues," *Int. Things*, vol. 12, no. 5, pp. 100289, 2020. doi: 10.1016/j.iot.2020.100289.
[14] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and SDN," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 3, pp. 1956–1971, 2021. doi: 10.1109/COMST.2021.3060582.
[15] SDX Central, "The future of network virtualization and SDN controllers," 2016. Accessed: Apr. 25, 2018. [Online]. Available: https://www.sdxcentral.com
[16] H. Galiza, M. Schwarz, J. Bezerra, and J. Ibarra, "Moving an IP network to SDN: A global use case deployment experience at AmLight," in *Brazilian Symp. Comput. Netw. Distrib. Syst. (SBRC) Workshop on Future Int. Res. Experiment. (WPEIF)*, Salvador Bahia Brazil, Jun. 2016, pp. 15–18.
[17] C. Macapuna, C. Rothenberg, and M. Magalhaes, "In-packet bloom filter based data center networking with distributed OpenFlow controllers," in *GLOBECOM Workshops (GC Wkshps)*, 2010, pp. 584–588.
[18] Open vSwitch, "Production quality, multilayer open virtual switch," Accessed: Jun. 6, 2024. [Online]. Available: http://www.openvswitch.org
[19] F. P. Deek and J. A. M. McHugh, *Open Source: Technology and Policy*. New York: Cambridge University Press, 2008.
[20] S. Walli, D. Gynn, and B. von Rotz, "The growth of open source software in organizations," 2005. Accessed: Apr. 25, 2018. [Online]. Available: http://www.optaros.com/en/publications/white_papers_reports
[21] M. Leary, "SDN, NFV and open source: The operator's view," 2014. Accessed: Apr. 25, 2018. [Online]. Available: http://research.gigaom.com/report/sdn-nfv-and-open-source-the-operators-view/
[22] I. H. S. Markit, "IHS Markit: 75% of carriers surveyed have deployed or will deploy SDN this year," The ComSoc Technology Blog website, 2016. Accessed: Apr. 25, 2018. [Online]. Available: http://techblog.comsoc.org/2016/09/08/ihs-markit-75-of-carriers-surveyed-have-deployed-or-will-deploy-sdn-this-year/
[23] Y. Han, "Software defined networking-based traffic engineering for data center networks," in *Proc. 16th Asia-Pacific Netw. Operat. Manag. Symp.*, Sep. 2014.
[24] ONF, "Protocol independent forwarding," Accessed: Apr. 25, 2018. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/OF-PIA_Protocol_Independent_Layer_for_OpenFlow_v1-1.pdf
[25] S. Rao, "SDN series part eight: Comparison of open source SDN controllers," 2018. Accessed: Apr. 25, 2018. [Online]. Available: http://thenewstack.io/sdn-series-part-eight-comparison-of-open-source-sdn-controllers/
[26] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature based comparison and selection of Software Defined Networking (SDN) controllers," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, Hammamet, Tunisia, Jan. 2014, pp. 1–7.
[27] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proc. 9th Central & Eastern Eur. Softw. Eng. Conf.*, Russia, ACM, 2013, pp. 1.
[28] J. Mccauley, "POX: A python-based OpenFlow controller," Accessed: Apr. 25, 2018. [Online]. Available: https://noxrepo.github.io/pox-doc/html/
[29] V. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, no. 3, pp. 105–110, 2008. doi: 10.1145/1384609.1384625.
[30] Nippon Telegraph and Telephone Corporation, "Ryu network operating system," 2012. Accessed: Apr. 25, 2018. [Online]. Available: http://osrg.github.com/ryu/
[31] Floodlight Is an Open SDN Controller, "Project floodlight," Accessed: Apr. 25, 2018. [Online]. Available: https://github.com/floodlight/floodlight
[32] OpenDaylight: Open Source Programmable Networking Platform, "OpenDaylight, a linux foundation collaborative project," Accessed: Apr. 25, 2018. [Online]. Available: http://www.opendaylight.org/software
[33] U. Krishnaswamy *et al.*, "ONOS: An open source distributed SDN OS," 2013. Accessed: Apr. 25, 2018. [Online]. Available: http://www.slideshare.net/umeshkrishnaswamy/open-network-operating-system

[34] Wiki ONOS Project, "Falcon release content of ONOS," 2018. Accessed: Apr. 25, 2018. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Falcon+release+content
[35] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of SDN controllers," in *Commun. (ICC) 2013 IEEE Int. Conf.*, 2013, pp. 3504–3508.
[36] M. Gerola *et al.*, "ICONA: Inter cluster ONOS network application," in *1st IEEE Conf. Netw. Softwar. (NetSoft)*, London, IEEE, 2015, pp. 1–2.
[37] T. Rosado and J. Bernardino, "An overview of Openstack Architecture," in *Proc. 18th Int. Database Eng. & Appl. Symp., IDEAS '14*, Porto Portugal, Jul. 2014, pp. 366–367. doi: 10.1145/2628194.2628195.
[38] C. Janz, L. Ong, K. Sethuraman, and V. Shukla, "Emerging transport SDN architecture and use cases," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 116–121, Oct. 2016. doi: 10.1109/MCOM.2016.7588279.
[39] M. Gerola, F. Lucrezia, M. Santuari, E. Salvadori, S. S. P. L. Ventre, and M. Campanella, "ICONA: A peer-topeer approach for software defined wide area networks using ONOS," in *Eur. Workshop Softw. Defined Netw. (EWSDN)*, Oct. 2016.
[40] A. Gupta *et al.*, "SDX: A software defined internet exchange," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, 2015. doi: 10.1145/2740070.2626300.
[41] M-CORD, "Mobile CORD," Accessed: Apr. 25, 2018. [Online]. Available: https://www.opennetworking.org/solutions/m-cord/
[42] International Telecommunications Union, Architecture of Optical Transport Networks, "ITU-T Recommendation G.872," 2019. Accessed: Apr. 25, 2018. [Online]. Available: https://www.itu.int/rec/T-REC-G.872-201912-S/en
[43] E-Cube, "E2 Infra, TMRND," Accessed: Feb. 27, 2018. [Online]. Available: http://www.tmrnd.com.my/research-themes/e-cube
[44] "Tech Mahindra joins ONOS as a collaborator," Accessed: Apr. 25, 2018. [Online]. Available: https://onosproject.org/2016/02/22/tech-mahindra-joins-onos-as-a-collaborator/
[45] W. Mitchell, "AARNET-X [PowerPoint slides]," 2017. Accessed: Apr. 25, 2018. [Online]. Available: https://meetings.internet2.edu/media/medialibrary/2017/10/11/20171017-mitchell-aarnet-x.pdf
[46] ESNet, GEANT, Indiana, U, and Internet2, "Perfsonar-performance service oriented network monitoring architecture," 2016. Accessed: Apr. 25, 2018. [Online]. Available: http://perfsonar.net
[47] GEANT, "Geant-european research and education network," Accessed: Jan. 19, 2024. [Online]. Available: http://www.geant.org
[48] S. Jain *et al.*, "B4: Experience with a globally-deployed software-defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 3–14, 2013.
[49] Y. H. Kim and D. Kim, "ONOS-based KREONET-S deployment and VDN application system," 2016. Accessed: Apr. 25, 2018. [Online]. Available: https://documents.pub/document/onos-based-kreonet-s-deployment-and-vdn-application-systemkreonet-snetreleaseonos-build-2016-yhkimpdf.html?page=1
[50] Wiki ONOS Project, "Global SDN deployment powered by ONOS," 2017. Accessed: Apr. 25, 2018. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Global+SDN+Deployment+Powered+by+ONOS
[51] Argela, "Argela-Secure Network Infrastructures," Accessed: Dec. 30, 2023. [Online]. Available: https://www.argela.com.tr/en/secure-network-infrastructures
[52] ONOS in action, "Onos project," 2017. Accessed: Apr. 25, 2018. [Online]. Available: https://onosproject.org/in-action/
[53] Huawei ICT Insights Publications, "China Unicom and Huawei Collaborate on SDN-ONOS [Press release]," Aug. 2016. Accessed: Jun. 21, 2024. [Online]. Available: https://e-file.huawei.com/~/media/EBG/Download_Files/Publications/en/ICT%20Insights%20Issue%2018.pdf
[54] Huawei ICT Insights Publications, "Enabling Service Innovation in Cloud Era [Press release]," September 2, 2016. Accessed: Apr. 25, 2018. [Online]. Available: https://www.huawei.com/en/press-events/news/2016/9/Industry-First-Full-Scenario-Agile-Controller-3
[55] Blue Planet Product, "Blue Planet ONOS," 2024. Accessed: Apr. 25, 2018. [Online]. Available: https://www.blueplanet.com/products

[56] Open CORD Project, "Open cord," 2018. Accessed: Apr. 25, 2018. [Online]. Available: https://opencord.org

[57] PR Newswire, "ECI develops an ONOS-based software-defined networking (SDN) controller [Press release]," Dec. 17, 2015. Accessed: Jun. 21, 2024. [Online]. Available: https://www.ecitele.com/media/1509/eci-collaborates-with-onos-press-release-dec-17-2015-final.pdf

[58] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. Hot-ICE "12 Proc. 2nd USENIX Conf. Hot Top. Manag. Internet, Cloud, Enterp. Netw. Serv.*, 2012, pp. 10.

[59] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 1–76, 2015. doi: 10.1109/JPROC.2014.2371999.

[60] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Surv. Tut.*, vol. 16, pp. 493–512, 2014.

[61] S. Azodolmolky, *Software defined networking with OpenFlow*. UK: Packet Publishing Ltd., 2013.

[62] O. Salman *et al.*, "SDN controllers: A comparative study," in *Elect. Conf. (MELECON), 2016 18th Mediterranean*, IEEE, 2016. Accessed: Apr. 25, 2018. [Online]. Available: https://www.researchgate.net/publication/304457462_SDN_controllers.

[63] J. Kim *et al.*, "OF@ TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," *Proc. Asia Pac. Adv. Netw.*, vol. 36, pp. 17–22, 2013. doi: 10.7125/APAN.36.3.

[64] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. Third Workshop on Hot Topics in Softw. Def. Netw., HotSDN '14*, New York, NY, USA, ACM, 2014, pp. 1–6.

[65] H. M. Noman and M. N. Jasim, "POX controller and open flow performance evaluation in software-defined networks (SDN) using mininet emulator," *IOP Conf. Series: Mat. Sci. Eng.*, vol. 881, pp. 012102, Jul. 2020.

[66] N. M. Kazi, S. R. Suralkar, and U. S. Bhadade, "Evaluating the performance of POX and RYU SDN controllers using mininet," in *Data Science and Comput. Intell.: Sixteenth Int. Conf. Inform. Process., ICInPro 2021, Oct. 22–24, 2021*, Bengaluru, India, Springer International Publishing. pp. 181–191.

[67] S. H. Choi and J. Kwak, "Performance analysis of simple IDS for floodlight controller in SDN Environment, international information institute (Tokyo)," *Information*, vol. 17, no. 11, p. 5435, 2014.

[68] M. A. M. Alrammahi and W. S. Bhaya, "Performance analysis for load balancing algorithms using POX controller in SDN," in *2022 Int. Conf. Data Science and Intell. Comput. (ICDSIC)*, IEEE, Nov. 2022, pp. 175–180.

[69] Y. Li, X. Guo, X. Pang, B. Peng, X. Li and P. Zhang, "Performance analysis of floodlight and Ryu SDN controllers under mininet simulator," in *2020 IEEE/CIC Int. Conf. Commun. in China (ICCC Workshops)*, IEEE, Aug. 2020, pp. 85–90.

[70] W. M. Othman, H. Chen, A. Al-Moalmi, and A. N. Hadi, "Implementation and performance analysis of SDN firewall on POX controller," in *2017 IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, IEEE, May 2017, pp. 1461–1466.

[71] S. Bhardwaj and S. N. Panda, "Performance evaluation using RYU SDN controller in software-defined networking environment," *Wirel. Pers. Commun.*, vol. 122, no. 1, pp. 701–723, 2022. doi: 10.1007/s11277-021-08920-3.

[72] A. H. Eljack, A. H. M. Hassan, and H. H. Elamin, "Performance analysis of ONOS and floodlight SDN controllers based on TCP and UDP traffic," in *2019 Int. Conf. Comput., Control, Elect., and Elect Eng. (ICCCEEE)*, IEEE, Sep. 2019, pp. 1–6.

[73] C. Fancy and M. Pushpalatha, "Performance evaluation of SDN controllers POX and floodlight in mininet emulation environment," in *2017 Int. Conf. Intell. Sustain. Syst. (ICISS)*, IEEE, Dec. 2017, pp. 695–699.

[74] R. C. Meena, M. Bundele, and M. Nawal, "RYU SDN controller testbed for performance testing of source address validation techniques," in *2020 3rd Int. Conf. Emerg. Technol. Comput. Eng.: Machine Learn Int. Things (ICETCE)*, IEEE, Feb. 2020, pp. 1–6.

[75] A. Sharma and G. S. Verma, "Performance comparison of Ryu and floodlight SDN controller," in *AIP Conference Proceedings*, AIP Publishing, Jun. 2023, vol. 2705, no. 1, pp. 040007. doi: 10.1063/5.0133684.
[76] R. K. Chouhan, M. Atulkar, and N. K. Nagwani, "Performance comparison of Ryu and floodlight controllers in different SDN topologies," in *2019 1st Int. Conf. Adv. Technol. Intelligent Control, Environ, Comput. & Commun. Eng. (ICATIECE)*, IEEE, Mar. 2019, pp. 188–191.
[77] D. Cabarkapa and D. Rancic, "Performance analysis of Ryu-POX controller in different tree-based SDN topologies," *Adv. Electr. Comp. Eng.*, vol. 21, no. 3, pp. 31–38, 2021. doi: 10.4316/AECE.2021.03004.
[78] I. Z. Bholebawa and U. D. Dalal, "Performance analysis of SDN/OpenFlow controllers: POX versus floodlight," *Wirel. Pers. Commun.*, vol. 98, no. 2, pp. 1679–1699, 2018. doi: 10.1007/s11277-017-4939-z.
[79] A. Singh, N. Kaur, and H. Kaur, "Extensive performance analysis of OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN controllers," *Microprocess. Microsyst.*, vol. 95, no. 15, pp. 104715, 2022. doi: 10.1016/j.micpro.2022.104715.
[80] S. Rowshanrad, V. Abdi, and M. Keshtgari, "Performance evaluation of SDN controllers: Floodlight and OpenDaylight," *IIUM Eng. J.*, vol. 17, no. 2, pp. 47–57, 2016. doi: 10.31436/iiumej.v17i2.615.
[81] M. T. Islam, N. Islam, and M. A. Refat, "Node to node performance evaluation through RYU SDN controller," *Wirel. Pers. Commun.*, vol. 112, no. 1, pp. 555–570, 2020. doi: 10.1007/s11277-020-07060-4.
[82] A. E. S. F. Ahmed and H. A. Elsayed, "Performance comparison of SDN wireless network under floodlight and POX controllers," in *2022 13th Int. Conf. Elect. Eng. (ICEENG)*, IEEE, Mar. 2022, pp. 91–95.
[83] A. B. Zainab and M. B. Al-Somaidai, "Network performance evaluation of distributed SDN using floodlight controllers," in *2019 2nd Int. Conf. Elect., Commun., Comput., Pow. Control Eng. (ICECCPCE)*, IEEE, Feb. 2019, pp. 34–39.
[84] I. K. Nisrina, A. Aginsa, E. Vinietta, and N. R. Syambas, "Link failure analysis in SDN data-center using RipL-POX controller," in *2015 9th Int. Conf. Telecommun. Syst. Serv. Appl. (TSSA)*, IEEE, Nov. 2015, pp. 1–6.
[85] E. Amiri, E. Alizadeh, and M. H. Rezvani, "Controller selection in software defined networks using best-worst multi-criteria decision-making," *Bull. Electr. Eng. Inform.*, vol. 9, no. 4, pp. 1506–1517, 2020. doi: 10.11591/eei.v9i4.2393.
[86] J. Ali and B. H. Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering," *Comput. Mater. Contin.*, vol. 67, no. 1, pp. 849–875, 2021. doi: 10.32604/cmc.2021.014576.
[87] M. Zaher and S. Molnár, "A comparative and analytical study for choosing the best suited SDN network operating system for cloud data center," *Annals of Emerg. Technol. Comput. (AETiC)*, vol. 6, no. 1, pp. 43–59, 2022. doi: 10.33166/AETiC.2017.10.01.
[88] B. Sapkota, B. R. Dawadi, and S. R. Joshi, "Controller placement problem during SDN deployment in the ISP/Telco networks: A survey," *Eng. Rep.*, vol. 6, no. 2, pp. e12801, 2024. doi: 10.1002/eng2.12801.
[89] J. Ali, B. H. Roh, and S. Lee, "QoS improvement with an optimum controller selection for software-defined networks," *PLoS One*, vol. 14, no. 5, pp. e0217631, 2019. doi: 10.1371/journal.pone.0217631.
[90] J. Ali, B. Lee, J. Oh, J. Lee, and B. H. Roh, "A novel features prioritization mechanism for controllers in software-defined networking," *Comput. Mater. Contin.*, vol. 69, no. 1, pp. 267–282, 2021. doi: 10.32604/cmc.2021.017813.
[91] D. Kannan and R. Thiyagarajan, "Entropy based TOPSIS method for controller selection in software-defined networking," *Concurr. Comput.: Pract. Exp.*, vol. 34, no. 1, pp. e6499, 2022. doi: 10.1002/cpe.6499.
[92] W. Kim, J. Li, J. W. K. Hong, and Y. J. Suh, "OFMon: OpenFlow monitoring system in onos controllers," in *2016 IEEE NetSoft Conf. Workshops (NetSoft)*, IEEE, 2016, pp. 397–402.
[93] S. A. R. Shah, S. Bae, A. Jaikar, and S. Y. Noh, "An adaptive load monitoring solution for logically centralized sdn controller," in *2016 18th Asia-Pacific Netw. Operat. Manag. Symp. (APNOMS)*, IEEE, 2016, pp. 1–6.
[94] H. Sahu, R. Tiwari, and S. Kumar, "SDN-based traffic monitoring in data center network using floodlight controller," *Int. J. Intell. Inform. Technol. (IJIIT)*, vol. 18, no. 3, pp. 1–13, 2022.

[95] Y. Han *et al.*, "Onvisor: Towards a scalable and flexible SDN-based network virtualization platform on ONOS," *Int. J. Netw. Manag.*, vol. 28, no. 2, pp. e2012, Mar./Apr. 2018. doi: 10.1002/nem.2012.

[96] C. M. Iurian, I. A. Ivanciu, B. M. Marian, D. Zinca, and V. Dobrota, "An SDN architecture for IoT networks using ONOS controller," in *2020 19th RoEduNet Conf.: Netw. Edu. Res. (RoEduNet)*, IEEE, Dec. 2020, pp. 1–6.

[97] Scott-Hayward, Sandra, "Trailing the Snail: SDN Controller Security Evolution," 2017. Accessed: Apr. 25, 2018. [Online]. Available: https://www.researchgate.net/publication/321241950_Trailing_the_Snail_SDN_Controller_Security_Evolution

[98] H. Gocher, S. Taterh, and P. Dadheech, "Impact analysis to detect and mitigate distributed denial of service attacks with Ryu-SDN Controller: A comparative analysis of four different machine learning classification algorithms," *SN Comput. Sci.*, vol. 4, no. 5, pp. 456, 2023. doi: 10.1007/s42979-023-01842-w.

[99] P. Ohri, S. G. Neogi, and S. K. Muttoo, "Security analysis of open source SDN (ODL and ONOS) controllers," in *2023 IEEE Int. Students' Conf. Elect., Electron Comput. Sci. (SCEECS)*, IEEE, Feb. 2023, pp. 1–4.

[100] J. Ma, R. Jin, L. Dong, G. Zhu, and X. Jiang, "Implementation of SDN traffic monitoring based on Ryu controller," *Int. Symp. Comput. Appl. Inform. Syst. (ISCAIS 2022)*, vol. 12250, no. 01, pp. 203–212, May 2022. doi: 10.1117/12.2639589.

[101] Y. Gautam, B. P. Gautam, and K. Sato, "Experimental security analysis of SDN network by using packet sniffing and spoofing technique on POX and Ryu controller," in *2020 Int. Conf. Network. Netw. Appl. (NaNA)*, IEEE, Dec. 2020, pp. 394–399.

[102] R. K. Arbettu, R. Khondoker, K. Bayarou, and F. Weber, "Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers," in *2016 17th Int. Telecommun. Netw. Strat. Planning Symp. (Networks)*, IEEE, Sep. 2016, pp. 37–44.

[103] M. Dagli, S. Keskin, Y. Yigit, and A. Kose, "Resiliency analysis of ONOS and Opendaylight SDN controllers against switch and link failures," in *2020 Fifth Int. Conf. Res. Comput. Intell. Commun. Netw. (ICRCICN)*, IEEE, Nov. 2020, pp. 149–153.

[104] Q. Ilyas and R. Khondoker, "Security analysis of floodlight, zeroSDN, beacon and POX SDN controllers," in R. Khondoker (ed.) *SDN and NFV Security*, Cham: Springer, 2018, vol. 30, pp. 85–98. doi: 10.1007/978-3-319-71761-6_6

[105] C. Bouras, A. Kollia, and A. Papazois, "Teaching network security in mobile 5G using ONOS SDN controller," in *2017 Ninth Int. Conf. Ubiquit. Future Netw. (ICUFN)*, IEEE, Jul. 2017, pp. 465–470.

[106] B. Sokappadu and A. Mungur, "A middleware for integrating legacy network devices into software-defined networking (SDN)," in *Towards New e-Infrastruct. e-Services for Develop. Countries: 12th EAI Int. Conf., AFRICOMM 2020*, Mauritius, Springer International Publishing, Dec. 2–4, 2020, pp. 121–139.

[107] A. Rajaratnam, R. Kadikar, S. Prince, and M. Valarmathi, "Software defined networks: Comparative analysis of topologies with ONOS," in *2017 Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, IEEE, Mar. 2017, pp. 1377–1381.

[108] Q. P. van, D. Verchere, H. Tran-Quang, and D. Zeghlache, "Container-based microservices SDN control plane for open disaggregated optical networks," in *2019 21st Int. Conf. Transparent Optical Netw. (ICTON)*, IEEE, Jul. 2019, pp. 1–4.

[109] R. Alvizu *et al.*, "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2232–2283, 2017. doi: 10.1109/COMST.2017.2715220.

[110] 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*, vol. 11, no. 17, pp. 2715, 2022. doi: 10.3390/electronics11172715.

[111] S. T. Arzo *et al.*, "MSN: A playground framework for design and evaluation of microservices-based sdn controller," *J. Netw. Syst. Manage.*, vol. 30, no. 1, pp. 1–31, 2022. doi: 10.1007/s10922-021-09631-7.

[112] D. Lake, N. Wang, R. Tafazolli, and L. Samuel, "Softwarization of 5G networks-implications to open platforms and standardizations," *IEEE Access*, vol. 9, pp. 88902–88930, 2021. doi: 10.1109/ACCESS.2021.3071649.
[113] M. H. Raza, S. C. Sivakumar, A. Nafarieh, and B. Robertson, "A comparison of software defined network (SDN) implementation strategies," *Procedia Comput. Sci.*, vol. 32, pp. 1050–1055, 2014. doi: 10.1016/j.procs.2014.05.532.
[114] ONF, "ONF demonstrates SMaRT-5G™ open source energy savings platform at Fyuz," Accessed: Dec. 12, 2023. [Online]. Available: https://opennetworking.org/news-and-events/press-releases/onf-demonstrates-smart-5g-open-source-energy-savings-platform-at-fyuz/
[115] Broadcom, "Software-defined networking open-source software innovation is driving more agile networks," Accessed: Dec. 12, 2023. [Online]. Available: https://www.broadcom.com/solutions/data-center/software-defined-networking
[116] The Linux Foundation, "Deutsche Telekom (DT) joins the Open Networking Foundation (ONF) as a Partner," Jul. 20, 2017. Accessed: Dec. 12, 2023. [Online]. Available: https://www.linuxfoundation.org/press/pressrelease/deutsche-telekom-dt-joins-the-open-networking-foundation-onf-as-a-partner
[117] D. Kurschner, SP360: Service Provider, "Cisco leads with 5G at mobile world congress 2019 Los Angeles," Nov. 4, 2019, Accessed: Dec. 12, 2023. [Online]. Available: https://blogs.cisco.com/sp/hitting-it-out-of-the-park-at-mobile-world-congress-2019-los-angeles
[118] S. M. Kerner, "Open source floodlight extends software defined networking to openstack," Jul. 10, 2012, Accessed: Dec. 12, 2023. [Online]. Available: https://www.enterprisenetworkingplanet.com/os/open-source-floodlight-extends-software-defined-networking-to-openstack/
[119] D. Ramel, "OpenDaylight SDN controller marks 6 years with 10th Release, Neon," Sep. 4, 2019. Accessed: Dec. 12, 2023. [Online]. Available: https://virtualizationreview.com/articles/2019/04/01/opendaylight-neon.aspx
[120] Menlo Park, "Coalition announces global breakthrough for software-defined networking," May 6, 2015. Accessed: Dec. 12, 2023. [Online]. Available: https://www.prnewswire.com/news-releases/coalition-announces-global-breakthrough-for-software-defined-networking-300078450.html
[121] A. Venkat, "Top 10 open source software risks for 2023," Mar. 1, 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://www.csoonline.com/article/574615/top-10-open-source-software-risks-for-2023.html
[122] Sunmait Technologies, "Choosing the right technology for your IT projects," Sep. 5, 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://medium.com/@sunmait/choosing-the-right-technology-for-your-it-projects-213feedcfe5c
[123] Help Net Security, "The double-edged sword of open-source software," Apr. 25, 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://www.helpnetsecurity.com/2023/04/25/open-source-dependencies
[124] S. S. Dang, "How to address cybersecurity issues in open source software?," Dec. 22, 2022. Accessed: Dec. 12, 2023. [Online]. Available: https://www.forbes.com/sites/sanjitsinghdang/2022/12/22/how-to-address-cybersecurity-issues-in-open-source-software/
[125] Expert Panel, Forbes Technology Council, "16 expert tips for choosing the best software vendor for your business," Apr. 1, 2022. Accessed: Dec. 12, 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2022/04/01/16-expert-tips-for-choosing-the-best-software-vendor-for-your-business
[126] L. Zhu *et al.*, "SDN controllers: A comprehensive analysis and performance evaluation study," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–40, 2020. doi: 10.1145/3421764.
[127] M. Aslam, D. Ye, M. Hanif, and M. Asad, "Machine learning based SDN-enabled distributed denial-of-services attacks detection and mitigation system for Internet of Things," in *Mach. Learn. Cyber Security: Third Int. Conf., ML4CS 2020*, Guangzhou, China, Springer International Publishing, Oct. 8–10, 2020, pp. 180–194.
[128] K. Alhamami and S. Albermany, "DDOS attack detection using machine learning algorithm in SDN network," in *2023 Al-Sadiq Int. Conf. Commun. Inform. Technol. (AICCIT)*, IEEE, Jul. 2023, pp. 97–102.

[129] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, "SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021. doi: 10.1109/ACCESS.2021.3101650.

[130] A. T. Kyaw, M. Z. Oo, and C. S. Khin, "Machine-learning based DDOS attack classifier in software-defined network," in *2020 17th Int. Conf. Elect. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, IEEE, 2020, pp. 431–434.

[131] V. Mohan, B. K. Madhavi, and S. B. Kishor, "Detection of UDP SYN flood ddos attack using random forest machine learning algorithm in a simulated software defined network," in *Int. Conf. Inform. Commun. Technol. Intell. Syst.*, Springer Nature Singapore, Apr. 2023, pp. 513–523.

[132] R. K. Chouhan, M. Atulkar, and N. K. Nagwani, "A framework to detect DDoS attack in Ryu controller based software defined networks using feature extraction and classification," *Appl. Intell.*, vol. 53, no. 4, pp. 4268–4288, 2023. doi: 10.1007/s10489-022-03565-6.

[133] C. Wang *et al.*, "METER: An ensemble DWT-based method for identifying low-rate DDoS attack in SDN," in *2021 IEEE 19th Int. Conf. Embed. Ubiquitous Comput. (EUC)*, IEEE, Oct. 2021, pp. 79–86.

[134] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and D. F. Carrera, "A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning," *J. Netw. Comput. Appl.*, vol. 205, no. 20, pp. 103444, 2022. doi: 10.1016/j.jnca.2022.103444.

[135] Y. Al-Dunainawi, B. R. Al-Kaseem, and H. S. Al-Raweshidy, "Optimized artificial intelligence model for DDoS detection in SDN environment," *IEEE Access*, vol. 11, pp. 106733–106748, 2023.

[136] K. Puranik *et al.*, "A two-level DDoS attack detection using entropy and machine learning in SDN," in *2023 3rd Int. Conf. Intell. Technol. (CONIT)*, Hubli, India, 2023, pp. 1–7. doi: 10.1109/CONIT59222.2023.10205776.

[137] H. Kousar, M. M. Mulla, P. Shettar, and D. G. Narayan, "Detection of DDoS attacks in software defined network using decision tree," in *2021 10th IEEE Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Bhopal, India, IEEE, 2021, pp. 783–788.

[138] A. K. Tahirou, K. Konate, and M. M. Soidridine, "Detection and mitigation of DDoS attacks in SDN using Machine Learning (ML)," in *2023 Int. Conf. Digital Age & Technol. Adv. Sustain. Develop. (ICDATA)*, IEEE, May 2023, pp. 52–59.

[139] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, "An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment," *Microprocess. Microsyst.*, vol. 89, no. 1, pp. 104412, 2022. doi: 10.1016/j.micpro.2021.104412.

[140] B. V. Karan, D. G. Narayan, and P. S. Hiremath, "Detection of DDoS attacks in software defined networks," in *2018 3rd Int. Conf. Comput. Syst. Inform. Technol. Sustain. Solutions (CSITSS)*, IEEE, Dec. 2018, pp. 265–270.

[141] S. Y. Mehr and B. Ramamurthy, "An SVM based DDoS attack detection method for Ryu SDN controller," in *Proc. 15th Int. Conf. Emerging Netw. Exper. Technol.*, 2019, pp. 72–73.

[142] J. A. Perez-Diaz, I. A. Valdovinos, K. K. R. Choo, and D. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," *IEEE Access*, vol. 8, pp. 155859–155872, 2020. doi: 10.1109/ACCESS.2020.3019330.

[143] B. Isyaku, K. B. A. Bakar, M. S. Ali, and M. N. Yusuf, "Performance comparison of machine learning classifiers for DDOS detection and mitigation on software defined networks," in *2023 IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS)*, IEEE, Jun. 2023, pp. 69–74.

[144] M. F. Akbaş, C. Güngör, and E. Karaarslan, "Usage of machine learning algorithms for flow based anomaly detection system in software defined networks," in *Intell. Fuzzy Tech.: Smart and Innov. Solutions: Proc. INFUS 2020 Conf.*, Istanbul, Turkey, Springer International Publishing, July 21–23, 2020, pp. 1156–1163.

[145] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "Security anomaly detection in software-defined networking based on a prediction technique," *Int. J. Commun. Syst.*, vol. 33, no. 14, pp. e4524, 2020. doi: 10.1002/dac.4524.

[146] L. M. Halman and M. J. Alenazi, "MCAD: A machine learning based cyberattacks detector in software-defined networking (SDN) for healthcare systems," *IEEE Access*, vol. 11, pp. 37052–37067, 2023. doi: 10.1109/ACCESS.2023.3266826.

[147] S. Sanagavarapu and S. Sridhar, "Route prediction in software defined networks using ensemble extreme learning machines," in *2021 Int. Conf. Inform. Commun. Technol. Converg. (ICTC)*, IEEE, Oct. 2021, pp. 1801–1806.

[148] R. Batra, M. Mahajan, and A. Goel, "An Optimized active learning TCM-KNN algorithm based on intrusion detection system," in *Congress on Intell. Syst.: Proc. CIS 2021*, Springer Nature Singapore, Jul. 2022, vol. 1, pp. 621–634.

[149] A. O. Alzahrani and M. J. Alenazi, "ML-IDSDN: Machine learning based intrusion detection system for software-defined network," *Concurr. Comput.: Pract. Exp.*, vol. 35, no. 1, pp. e7438, 2023. doi: 10.1002/cpe.7438.

[150] A. Ashraf, Z. Iqbal, M. A. Khan, U. Tariq, S. Kadry and S. O. Park, "Scalable offloading using machine learning methods for distributed multi-controller architecture of SDN networks," *J. Supercomput.*, vol. 78, no. 7, pp. 10191–10210, 2022. doi: 10.1007/s11227-022-04313-w.

[151] C. Manso *et al.*, "First scalable machine learning based architecture for cloud-native transport SDN controller," in *Optical Fiber Commun. Conf.*, Optica Publishing Group, Jun. 2021, pp. F4H-6.

[152] S. Liang, W. Jiang, F. Zhao, and F. Zhao, "Load balancing algorithm of controller based on SDN architecture under machine learning," *J. Syst. Sci. Inform.*, vol. 8, no. 6, pp. 578–588, 2020. doi: 10.21078/JSSI-2020-578-11.

[153] G. Choudhury, G. Thakur, and S. Tse, "Joint optimization of packet and optical layers of a core network using SDN controller, CD ROADMs and machine-learning-based traffic prediction," in *Optical Fiber Commun. Conf.*, Optica Publishing Group, Mar. 2019, pp. M2A-1.

[154] G. Ramya and R. Manoharan, "Prediction based dynamic controller placement in SDN," *EAI Endorsed Transact. Scalable Inform. Syst.*, vol. 8, no. 32, pp. 1–14, Apr. 2021. doi: 10.4108/eai.27-4-2021.169420.

[155] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine learning and deep learning based traffic classification and prediction in software defined networking," in *2019 IEEE Int. Symp. Measure. & Netw. (M&N)*, IEEE, Jul. 2019, pp. 1–6.

[156] A. Malik, R. de Fréin , M. Al-Zeyadi, and J. Andreu-Perez, "Intelligent SDN traffic classification using deep learning: Deep-SDN," in *2020 2nd Int. Conf. Comput. Commun. Inter. (ICCCI)*, IEEE, Jun. 2020, pp. 184–189.

[157] A. I. Owusu and A. Nayak, "An intelligent traffic classification in SDN-IoT: A machine learning approach," in *2020 IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, IEEE, 2020, pp. 1–6.

[158] J. Suárez-Varela and P. Barlet-Ros, "Flow monitoring in software-defined networks: Finding the accuracy/performance tradeoffs," *Comput. Netw.*, vol. 135, no. 2, pp. 289–301, 2018. doi: 10.1016/j.comnet.2018.02.020.

[159] T. T. Huong, N. D. D. Khoa, N. X. Dung, and N. H. Thanh, "A global multipath load-balanced routing algorithm based on reinforcement learning in SDN," in *2019 Int. Conf. Inform. Commun. Technol. Converg. (ICTC)*, IEEE, Oct. 2019, pp. 1336–1341.

[160] S. K. Tayyaba *et al.*, "5G vehicular network resource management for improving radio access through machine learning," *IEEE Access*, vol. 8, pp. 6792–6800, 2020. doi: 10.1109/ACCESS.2020.2964697.

[161] R. P. Singh, J. Grover, and G. R. Murthy, "Self organizing software defined edge controller in IoT infrastructure," in *Proc. 1st Int. Conf. Int. Things and Mach. Learn.*, Oct. 2017, pp. 1–7.