Journal of Network and Computer Applications 174 (2021) 102856

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A comprehensive survey of load balancing techniques in software-defined network

Mosab Hamdan [a,*], Entisar Hassan [a], Ahmed Abdelaziz [c], Abdallah Elhigazi [b], Bushra Mohammed [a], Suleman Khan [d], Athanasios V. Vasilakos [e,**], M.N. Marsono [a,***]

[a] School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Johor, Malaysia
[b] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor, Malaysia
[c] Faculty of Computer Science, Future University, Khartoum, Sudan
[d] Department of Computer and Information Sciences, Northumbria University, Newcastle Upon Tyne NE1 8ST, UK
[e] School of Electrical and Data Engineering, University of Technology Sydney, New South Wales, NSW 2007, Australia

ARTICLE INFO

Keywords:
Load balancing
Software-defined network
Multiple controller
Distributed controller

ABSTRACT

A software-defined network (SDN) separates the network control plane from the data forwarding plane. SDN has shown significant benefits in many ways compared to conventional non-SDN networks. However, traffic distribution in SDN impacts efficiency and raises many other challenges. For instance, uneven load distribution in the SDN significantly impacts the network performance. Hence, several SDN load balancing (LB) techniques have been introduced to improve the efficiency of SDN. In this article, we provide a thematic taxonomy of LB in SDN, considering several parameters from the past technical studies such as the objectives of LB, data plane LB techniques, control plane LB techniques, other aspects of data plane/control plane LB as well as the performance metrics for LB techniques. Furthermore, useful insights on LB and a comparative analysis of various promising SDN LB techniques are also included in the survey. Finally, existing challenges and future direction on SDN LB techniques are highlighted.

1. Introduction

In the past decade, network requirements have rapidly changed in response to the escalating size of network traffic and quality requirements; thereby, putting more demand on end-to-end goals. Conventional network architectures are static and, thus, complex to address dynamic network conditions. In order to allow networks to be adaptive needs, an emerging new network model termed software-defined networking (SDN) has been explored (Chica et al., 2020; Latif et al., 2020; Nayyer et al., 2019a; Li et al., 2016). In essence, SDN separates the network control plane from that of the data forwarding plane (McKeown et al., 2008a). As an example, the data forwarding layer uses OpenFlow switches that are programmable via the OpenFlow controller. These switches use a southbound protocol application program interface (API) (e.g., the OpenFlow protocol) for communication with the controller (OpenFlow Protocol v1.5 and 2, 2014). The SDN controller is responsible for the data plane devices (router and switch), which centrally defines the network policy topology and manages multiple interfaces southbound protocols (Feamster et al., 2013).

The OpenFlow protocol (McKeown et al., 2008b) provides solutions that are effective and efficient for monitoring network traffic and to provide elastic topology. It permits software that runs on multiple routers and facilitates their path association with the packet via the network (Shirmarz and Ghaffari, 2020; Gude et al., 2008). Conventional non-SDN networks are not capable of providing a global view of network topology and resources. Thus, load balancing (LB) mechanisms are not explicitly defined (Devapriya and Gandhi, 2020; Zeng et al., 2019; Long et al., 2013). Consequently, due to the growing interest in SDN and its potentials of replacing conventional networks in addition to the industry concerns, LB is considered an important open issue (Lin et al., 2016c). SDN can ensure a simplified network management system while realizing the invention and development of computer networks. Since the network resources, and requisite knowledge of application for optimizing the load, are known via the controller, it is suitable to implement

* Corresponding author.
** Corresponding author.
*** Corresponding author.
E-mail addresses: Mosab.hamdan@ieee.org (M. Hamdan), th.vasilakos@gmail.com (A.V. Vasilakos), mnadzir@utm.my (M.N. Marsono).

https://doi.org/10.1016/j.jnca.2020.102856
Received 6 June 2020; Received in revised form 4 September 2020; Accepted 30 September 2020
Available online 24 October 2020
1084-8045/© 2020 Elsevier Ltd. All rights reserved.

LB. As a result, SDN brings new possibilities to improve the balance of technology in the conventional network load. Today, the LB solution in SDN is effective, but the flexibility in customization remains limited. Typically, in a cloud environment, a service provider hosts various types of services and applications via a multi-tenant service that requires a specific LB scheme. Therefore, it is not easy to customize the LB system as different load balancers may be required for each of the services provided, which may be too expensive to be widely used (Shang et al., 2013).

A centralized control model is used by the OpenFlow protocol (Shah et al., 2013; Othman and Okamura, 2013). This consists of controllers, switches, and protocols (Shirmarz and Ghaffari, 2020; Shah et al., 2013). The controller selects all network routes in a centralized control model, and packets that appear first in the individual data stream is conveyed to the SDN controller. The controller then carries out the computation of the routing path for individual data streams through configuring switch's flow table continuously through observability of the global network view. Usually, the first packet, which is related to each data stream becomes the most needed for initialization request. However, the capacity associated with the request processing of a sole controller is inadequate. For example, the NOX controller (Tootoonchian et al., 2012) can handle about 30,000 requests per second, while the Maestro controller (Ng et al., 2010) can handle around 600,000 requests per second. Due to this constraint in request handling, the control channel may become congested or overloaded (Alsaeedi et al., 2019; Ma et al., 2017; Phemius and Thales, 2013). As with traditional networks, network congestion problems can be resolved through key link redundancy in SDN, which also provides network robustness and stability. Through the process of ensuring efficient traffic distribution among numerous paths in SDN, LB objectives can be attained. Achieving this goal has become a major concern for network researchers and managers over the years. Hence, the need to explore load balance research in SDN has become important.

Although there exist several impressive surveys in SDN LB such as (Belgaum et al., 2020; Semong et al., 2020; Mehra et al., 2019; Neghabi et al., 2018; Kumari and Thakur, 2017; Badirzadeh and Jamali, 2018; Zhang et al., 2018; Li and Xu, 2017), this paper differs in that it classifies and analyzes LB in SDN from a different perspective. For instance, Kumari et al. (Kumari and Thakur, 2017), Li et al. (Li and Xu, 2017), Badirzadeh et al. (Badirzadeh and Jamali, 2018), Zhang et al. (2018), and Mehra et al. (2019) surveyed different aspects of LB, especially relating to SDN in the data center. However, none of these works considered aspects of LB in SDN networks. In another study, Neghabi et al. (2018) presented a review on LB mechanisms used in the SDN, focusing on two categories of LB approaches:deterministic and non-deterministic. However, the drawback associated with their approach is that the parameters used in their investigation are not sufficient to justify their LB scheme's performance. That has become a significant source of concern, as this could cause load performance issues in terms of LB. Hence, this paper sheds light on the challenges, research opportunities and limitations of SDN LB techniques. At the same time, most of these surveys focused on LB schemes in the software-defined data centers.

Systematic analysis of LB techniques and algorithms used by various researchers was proposed in (Belgaum et al., 2020). The selected papers are classified into artificial intelligence and traditional LB based approaches according to the method used to address SDN LB challenges. Similarly, this work addresses the problems that have been discussed, the approaches used, and the solutions that have been proffered. Based on different studies considered in this work, we discovered that numerous techniques did not meet specific critical requirements that it is essential to increase the effectiveness of the current SDN LB methods. Furthermore, Semong et al. (2020) offer a summary of the available LB schemes in SDN. The review focused on research issues, current approaches, and future research directions for interested researchers. A description of emulators/mathematical tools regularly utilized in designing SDN algorithms for smart LB was provided. A description of the efficiency metrics used to test the algorithms was also given by evaluating various methods to study pertinent literature on intelligent LB for SDN. However, none of the existing surveys has given the proper taxonomy for the categorization of different LB techniques in SDN. Therefore, our paper performs a thorough analysis with in-depth information compared to earlier works. The taxonomy of this work covers various aspects of LB solutions such as the objectives of LB, data plane LB techniques, control plane LB techniques, other aspects of LB in SDN, and performance metrics used for LB techniques, and other aspects of SDN LB. A summary and a comparative analysis of diverse and promising SDN LB techniques are also included in the survey for the curious reader.

Additionally, this paper sheds light on the challenges, research opportunities, and limitations of SDN LB techniques. Table 1 provides a list of survey works on SDN LB topics and their respective targets. The table columns show the survey papers and year, taxonomy, data plane types LB, control plane types LB, other aspects of data/control planes LB (other aspects of D/CP LB), comparative study, issues, and challenges. The symbol ✓ indicates whether the subject is covered in the respective survey.

1.1. Scope and contribution

Several load balancing techniques in SDN have been recently introduced with a particular emphasis on two aspects of SDN LB techniques: data plane and control plane LBs. This article aims to provide an overview of LB in SDN through classification and analyses of issues related to the research area while further exposing approaches necessary for future study. The contributions of this article are as follow:

1. This paper provides a comprehensive literature review of pertinent studies on LB in SDN. The selection criteria of these studies were based on citation count, the year of publication, merit, and published in peer-reviewed and high-quality journals and conferences over the last decade; especially related studies on data and control planes in LB.
2. This paper introduces a thematic taxonomy for classifying current SDN LB literature into a set of categories.
3. This paper analyzes current SDN LB techniques based on the proposed thematic taxonomy.
4. Finally, this survey focuses on the most recent research in SDN LB techniques, pointing out the key challenges that require attention as future research opportunities. These opportunities would help and motivate researchers for future work in SDN LB.

1.2. Paper organization

This survey elaborates and discusses the related issues associated with each aspect of the topic. A brief discussion of the need for LB in SDN and description of the SDN architecture is presented in Section 2. Section 3 discusses LB's taxonomy in SDN, which are divided into objectives of LB, data plane LB, control plane LB, other aspects of data/control planes LB, and performance metrics for LB techniques. Meanwhile, Section 4 presents a discussion and a comparative analysis of LB techniques in SDN. Furthermore, a summary of our survey findings is presented in Section 5. Section 6 summarizes the challenges and open areas of further research on SDN LB techniques, followed by a conclusion in Section 7.

2. Overview of SDN load balancing

A brief overview of SDN architecture and the importance of the SDN LB are presented in this section.

2.1. SDN architecture

The SDN architecture is associated with three main components:

Table 1
Comparison of survey works on SDN load balancing.

Survey Paper	Year	Taxonomy	Types of Data Plane LB	Types of Control Plane LB	Other Aspects of D/CP LB	Comparative Analysis	Issues and Challenges
Kumari et al. (Kumari and Thakur, 2017)	2017	×	✓	✓	×	×	×
Li et al. (Li and Xu, 2017)	2017	✓	✓	✓	×	×	×
Badirzadeh et al. (Badirzadeh and Jamali, 2018)	2018	×	✓	✓	×	×	×
Zhang et al. (Zhang et al., 2018)	2018	✓	✓	×	×	×	✓
Neghabi et al. (Neghabi et al., 2018)	2018	✓	×	✓	×	✓	✓
Mehra et al. (Mehra et al., 2019)	2019	×	×	✓	×	×	×
Semong et al. (Semong et al., 2020)	2020	×	✓	×	×	×	✓
Belgaum et al. (Belgaum et al., 2020)	2020	✓	×	✓	×	✓	✓
Our survey	2020	✓	✓	✓	✓	✓	✓

control plane that consists of one or multiple controllers; data plane that is characterized by the presence of network equipment (e.g., routers, switches, as well as middleboxes) that interact to form a data forwarding network; and the SDN application layer that provides an avenue for executing all network applications. The architecture of SDN is depicted in Fig. 1. One of SDN architecture's major characteristics is its ability to separate control and data layers from other layers. Another key feature of SDN is programmability, which permits users to configure their application specifically within the layer associated with the application based on the north-bound interface that offers programmable APIs and highly sophisticated policy applications and services. Besides, the OpenFlow protocol offers a standard API delivered via the south-bound interface (Akyildiz et al., 2014; Bozakov and Sander, 2013; McKeown et al., 2008b). The SDN controller plays the role of a network operating system, which views the comprehensive network topology as well as ensuring a secure line of communication between OpenFlow switches (Clayman et al., 2016). It is also responsible for managing, controlling, and manipulating the flow table within the switch. The north-bound and south-bound interfaces are the primary communication interfaces of the SDN controller (Kuźniar et al., 2015). The SDN technology uses some well-known protocols, such as OpenFlow, to ensure smooth communication between the control and infrastructure layers (McKeown et al., 2008b).

The key features of SDN include:

1. Separation and abstraction of control and data planes.
2. Intelligence is logically centralized and thus has a global view of the network and changing needs.
3. Possibility of creating various applications through the application of the underlying network infrastructure.
4. Programmability of data plane to ensure simplicity and versatility in networking.
5. Use of accelerated innovation to speed up and promote business innovation and allow IT and network operators to implement programs in real-time. This to ensures that the network has been reprogrammed to satisfy both business and customer demands.

The network architecture and its separation from individual network resources can be based on the virtualization process (Jimson et al., 2019; Fundation, 2012; Bera et al., 2017). The main reason behind the performance of SDN network is due to the SDN controller, which is responsible for controlling the infrastructure layer function, and the switches that are converted into high-speed packet processors and packet forwarding actuators. The communication between packet forwarder and controller is carried out using protocols of the control signaling like OpenFlow, which is the most commonly used (OpenFlow

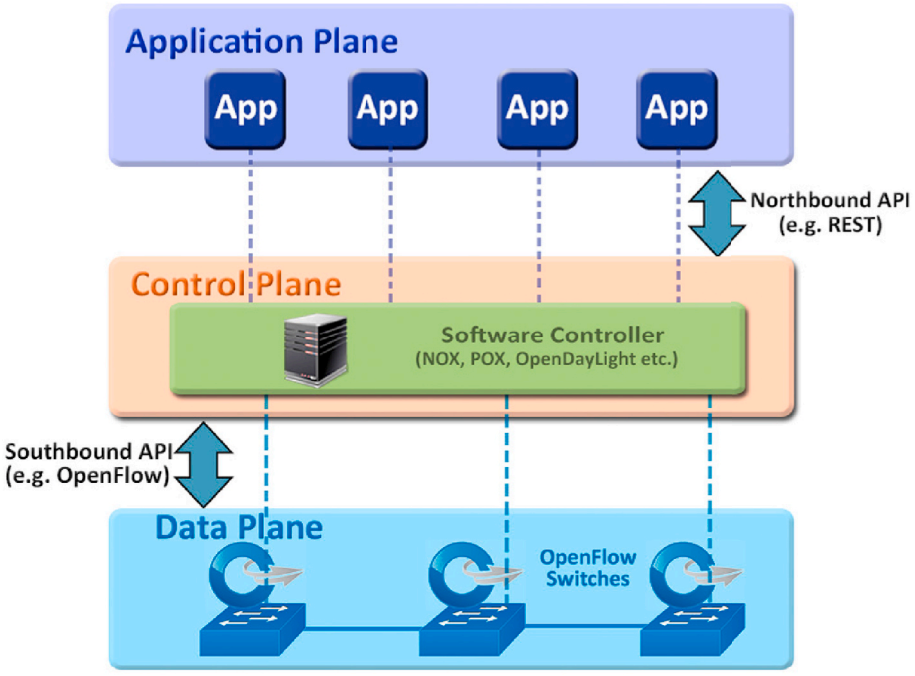

Fig. 1. SDN architecture (Singh and Bhandari, 2020).

Protocol v1.5 and 2, 2014).

2.2. Importance of SDN load balancing

Network management can be simplified in SDN while realizing innovation and evolution in computer networks. Besides, the fact that the controller can view network resources globally coupled with the knowledge requirements of applications for load optimization makes it suitable for SDN LB. As a result, SDN brings new possibilities to improve the balance of technology in the conventional network load.

Furthermore, LB technology has been crucial to the SDN networks to improve its performance in multiple aware routing approaches (Kumari and Thakur, 2017). It has also been employed to effectively allocate the resources of the network for the overall improvement of the performance of the network and quality-of-service (QoS) (Li and Xu, 2017). Therefore, the overall performance can be enhanced through the use of LB technology (Zhou et al., 2010; Salman et al., 2014; Akbar Neghabi et al., 2019). A load balancer based on SDN enables the control of several computers; thus, ensuring the possibility of a more agile network. The ability to directly configure the network ensures improved network management for more flexible and effective application services (Kaur et al., 2018). Although computing and storage have seen advancement in virtualization and automation, networks are lagging behind. LB using SDN enables the network to act like the virtualized computing and storage models. LB in SDN helps in the exploration of the best route and application for a faster request delivery (Neghabi et al., 2018).

3. Taxonomy of load balancing in SDN

In this section, we provide a thematic taxonomy for LB techniques in SDNs. Fig. 2 classifies existing SDN LB solutions following a set of standard parameters, as in the majority of related literature. The parameters selected for this thematic taxonomy were constructed from six factors: objectives of LB, data plane LB techniques, control plane LB

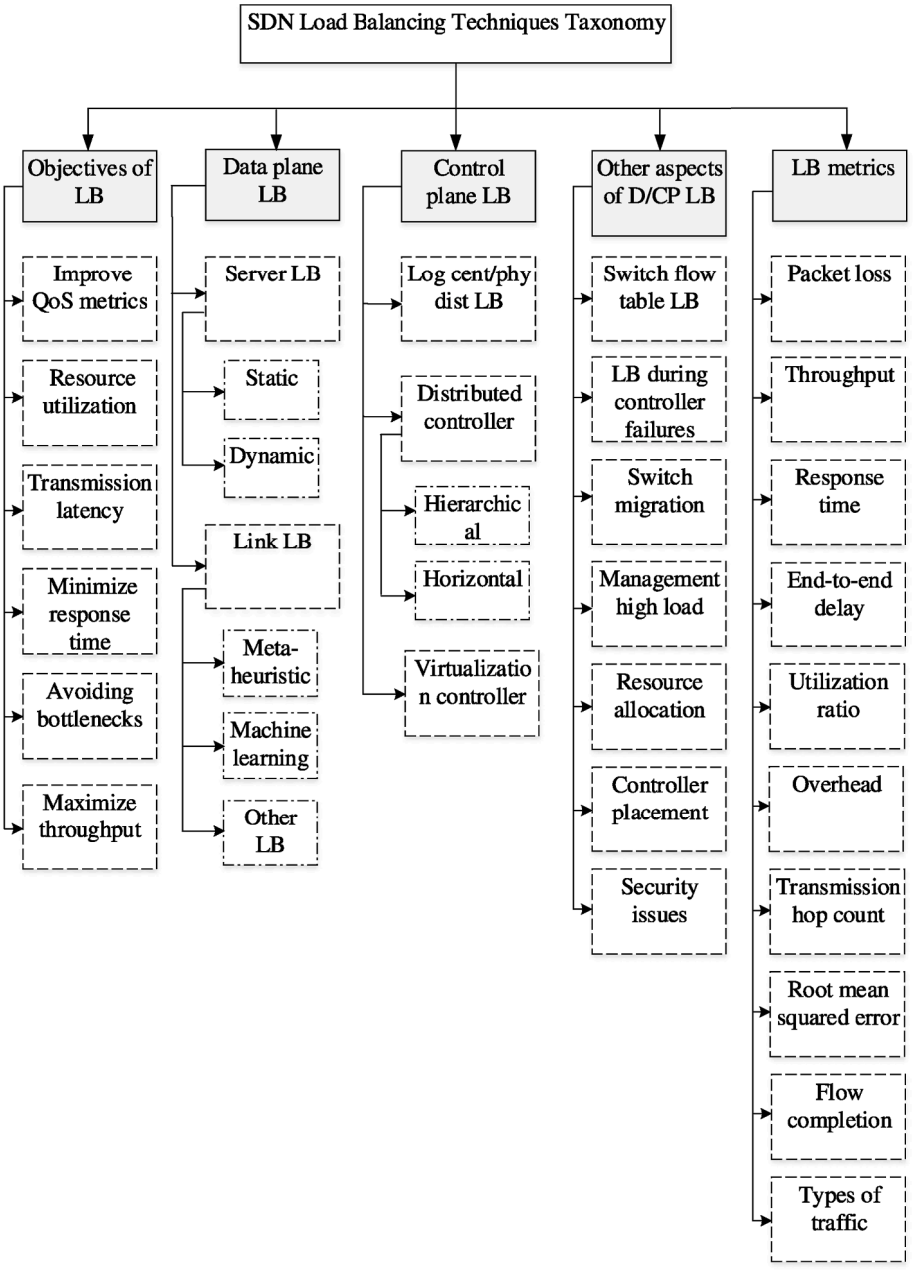

Fig. 2. Thematic taxonomy of SDN load balancing techniques.

techniques, other aspects of data plane/control plane LB (other aspects of D/CP LB), and performance metrics used for LB techniques. Each of these factors is comprehensively discussed in the following subsections.

3.1. Objectives of load balancing

The goals of SDN LB are to boost the QoS metrics, maximize throughput, maximize response time, avoid congestion, and optimize the usage of resources. These SDN LB objectives are discussed as follow:

1. Improve end-to-end QoS metrics

The primary aim of the LBs is to ensure end-to-end QoS for the SDN network. Therefore, improving the efficiency and overall performance of the system. The QoS aims to provide a better user experience by preventing excessive device latency, optimizing performance, and reducing response time (Šeremet and Čaušević, 2020; Karakus and Durresi, 2017a; Boero et al., 2016).

2. Optimize resource utilization

This is one of the overarching goals of LB since the proper use of resources is important for the efficiency of the SDN LB model. Hence, there is a degree to which network resources such as link, bandwidth, processor, and memory usage are to be utilized. A suitable resource provision algorithms ensure the maximal usage of resources for the LB (Hu et al., 2017b; Adami et al., 2014; Rupani et al., 2020).

3. Reduce transmission latency

The latency of transmission refers to the time it takes the host switch to transmit data. This depends on several factors that include the switches performance, whether the transmission queue is congested and the data packets' size. The transmission latency shows congestion in the link, and in some other way, it also serves as the switch load condition. Thus, the SDN controller must collect the bytes that are transmitted within a specific period and the transmission rate. This parameter should be reduced (Belgaum et al., 2020; Neghabi et al., 2018; Chen--Xiao and Ya-Bin, 2016).

4. Minimize response time

This is specified by the time interval between the time a server request is received, and the time it is answered, or mission accomplished. Therefore, in a distributed SDN network, it is time it takes a particular LB algorithm to react. This parameter should be minimized (Zhong et al., 2018; Jamali et al., 2019).

5. Avoiding bottlenecks

To avoid any congestion or bottlenecks in the SDN network setting, LB methods are required to spread the load equally between different switches/controllers so that no switch/controller gets overloaded (i.e., among the bottlenecked switches of each link, the best is the one with the lowest capacity). Proper LB can reduce resource consumption through efficient use of the available resources. Additionally, it enforces failover, allows scalability, prevent bottlenecks, and reduce response time (Jamali et al., 2019; CN et al., 2019; Shi et al., 2020).

6. Maximize the throughput

High throughput is a desirable attribute needed for a high-performance network that is only feasible if the workload and resources are spread equally to the various nodes. This is the size of correctly transferred data over a certain period of time from one location to another (Daraghmi and Yuan, 2015; Ramya et al., 2020; Liu et al., 2019).

3.2. Load balancing for data plane (LB for DP)

This technique is used to attain LB with low latency network performance, especially within the data plane. It is also used to solve the load imbalance in paths and severs and avoid network bottlenecks in SDN. Data plane LB can be classified as servers LB and links LB are discussed in what follows.

3.2.1. Server load balancing

The current network traffic is extremely large and still increasing. However, for service providers, it has become a very serious problem as it results in network congestion and server overload. Grand View Research (Research and G.V, 2017), which is a market research company, reported that traffic in the data center is expected to reach 9965 Exabyte by 2020 as compared to 4515 Exabyte in 2015, and market penetration of SDN-based data center traffic is expected to increase exponentially from over 58% to over 75% by 2020. The increased traffic emphasizes the need to balance the number of LB services needed to increase the amount of traffic on the SDN (Research and G.V, 2017). Another study by Microsoft research labs compared the Internet traffic ratio between eight data centers and services over a week. The study found that 44% of the total traffic was a virtual IP address (VIP), representing LB traffic or source network address translation (SNAT), 30% of service traffic, and 26% Internet traffic (Patel et al., 2013).

Given that navigating through the load balancer is required for a bulk of the traffic, 70% of the VIP traffic is shown by this study to be managed by a single data center. Hence, the load balancer is often located behind a firewall, handling all VIP traffic (Poddar et al., 2015). An LB strategy can be used to distribute traffic to different servers to overcome network congestion. On the other hand, the server LB techniques are divided into two parts: static (Leland and Hendrickson, 1994) and dynamic (Shabtay, 2010).

3.2.1.1. Static algorithms for server load balancing. Static LB algorithms are simple but costly. They are suitable for homogeneous servers, but their inflexible nature makes them unsuitable for dynamic changes (LD and Krishna, 2013). The static algorithm performs load distribution with little or no consideration for the efficiency of the component nodes such as RAM size, server processor, and the link bandwidth. Nevertheless, this algorithm has little overhead with easy implementation, with less overhead and suitable for homogeneous servers. This algorithm is also not sufficiently versatile to cater to complex changes to the attributes. For example, more tasks are intermittently sent to the same server without consideration of the server's ability to handle such task size at the time.

The first LB application in SDN based on the OpenFlow switch was introduced in (Uppal and Brandon, 2010). In this architecture, a static IP address is used in each server alongside a web server emulator running on a specific port. The NOX controller manages the server list. As new requests are made from the client, the OpenFlow switch checks the flow table for matching entries. An LB module written in the C++ language was developed on the NOX controller to provide LB services. However, this study only allows a single controller to manage a switch. Two LB algorithms-static and dynamic scheduling algorithms, designed on the SDN network architecture were presented in (Zhang and Guo, 2014).EE E The experiment results showed that the dynamic algorithm efficiency is higher than that of the static algorithm. In (Kaur et al., 2015), the authors implemented and compared a round-robin LB strategy with a random strategy already implemented using an OpenFlow switch connected to a POX controller.

3.2.1.2. Dynamic algorithms for server load balancing. Dynamic LB algorithm allocates the load according to the current state of the network

node (Alakeel, 2010). The LB algorithm carries out a routine check on the link capacity and server's load at run-time. Implementing server LB for dynamic algorithms is simplified by the use of SDNs, which permits the programmability and flexibility of network appliances (Raghul et al., 2017).

In order to balance the load in the server pool, a genetic algorithm (GA) was implemented in (Chou et al., 2014), to redirect the flow to achieve optimal LB effectively. This study assumes that there are $\mathbb{N}$ flows, each with a different load. There is a varying amount of work for every server within the server pool. For the coefficient of the server to be minimized, a fitness function was proposed. There are two main components related to the architecture: the OpenFlow switch and the OpenFlow controller. Three modules are built on the controller; a flow control module, a decision module, and a monitoring module. The OpenFlow switch components include three modules, namely; OpenFlow handler, a stream modification, and packet mirroring modules. Four LB algorithms were compared namely: a load-based, round-robin, random and genetic algorithm. Compared, with other algorithms, the genetic-based algorithm showed a significantly better result. The entire experimental setup was in a simulated environment with a simple topology. However, the architecture should be tested in a real environment with a large network topology to assess its scalability and stability.

Another LB approach is the dynamic aware LB system that uses a Floodlight controller. A server-based load balancer (SBLB) technique was proposed, and a comparative analysis was carried out with two other techniques, random, and round-robin, in (Shang et al., 2013). The controller chooses the smallest load associated with a server based on the weight on the minimum connection of the selected node regarding the number of active connections. The load on the server is obtained by dividing, the number of active connections by the servers' weight. The experiments were carried out in Mininet simulations (Mininet simulation and 2019., 2019). In another work, Chen et al. (2015a) proposed dynamic LB technology in a virtualization environment based on OpenFlow. This method was connected to the controller and server cluster by the Open Flow switch network, which connects to the Open Flow switch network while providing a storage-area network (SAN). It also scans most of the clusters that offer shared storage for the server cluster. Compared to round-robin and random algorithms, the SBLB algorithm enhanced server CPU efficiency, memory utilization, and response time. However, this method is deficient in that it cannot perform real-time monitoring of the load in addition to using reactive flow entry.

Du et al. (Du and Zhuang, 2015) introduced an algorithm called dynamic server LB; using the sFlow traffic monitoring tool (http://www.sFlow.org/sFlo, 2018) and OpenFlow to distribute traffic among servers of the cluster effectively. This method uses the wildcard rules to combine the traffic of the server replicas and makes decisions according to statistics of real-time traffic collected through the sFlow. Moreover, the switch's wildcard rules allow sending requests to many clients and ignoring the SDN controller, thereby decreasing the size of the rules and the network's latency. The authors developed the algorithm as a Floodlight controller application module and evaluated its effectiveness with Mininet simulation. Compared to random and round-robin algorithms, this algorithm enhanced the throughput, reduced the latency, and balanced the server load. However, this paper has some limitations, such as the small size of the network and its use of software switch in a virtual environment.

Moreover, Zhong et al. (2017a) presented an effective dynamic SDN LB method to addressservice response time as their most important factor for determining the experience of the user within the model provided by the service. This model involves the server clusters. This method presented a server response time based (LBBSR) dynamic LB scheme in the SDN architecture. The SDN controller obtains each server's response time and selects the one with the least or most stable time of response. This scheme leverages server resources, compared with that of the conventional round-robin and random schemes, for an improved LB performance. This approach is more cost-effective than traditional solutions because of the reduced hardware requirements and software customization method. Although this scheme effectively overcomes many LB issues, it fails to consider energy savings in server LB.

Wang et al. (WANG et al., 2016) introduced a dynamic SDN load balance scheme (SDN-LB) for cloud data centers and addressed LB's issue through the deployment of the SDN architecture. The researchers utilized the Plug-n-Server (Handigol et al., 2009a) to implement the LB over an unstructured networks method called LOBUS in the SDN architecture. The three parts associated with the plug-in server include the objective underlying composite by servers and clients; decision platform and the SDN controller OpenFlow network switch. There are four modules associated with the controller: dynamic load scheduling, traffic detection, flow management, and load calculation. The traffic detection module dynamically monitors the traffic statistics; the load calculation module estimates the load distribution of the cloud environment. ExAbd-Allah et al., 2019perimental results have shown that the LOBUS achieved better throughput than the oblivious and stateful LB methods. This is because of the single controller used in the benchmarked algorithms, which results in system bottleneck as well as low scalability and availability. Moreover, conventional or benchmarked algorithms have not considered latency and utilization metrics. Similarly, there is no measurement based on the uniformity of the load distribution among servers.

Abdelltif etal., 2018 proposed a cloud server LB service (SBLB) focused on the SDN to optimize resource efficiency and reduce user response time. The proposed technique includes an application module that runs over an SDN controller and the servers that use OpenFlow switches to connect the controller. The Framework module comprises a device management module, a dynamic LB module, and a control module. The controller handles all communications, controls host pools, and keeps a host charging in real-time. Experimental findings confirm the efficiency of the proposed scheme.

Furthermore, an LB algorithm was proposed in (Srivastava and Pandey, 2020). This algorithm considered network as a graph whose edges and vertices are the network channels and switches. This approach calls for the superiority of the bandwidth of the channel over the server's load to determine the best path dynamically; this claim improves the QoS parameters in terms of network stability and efficiency. The proposed algorithm functions in two stages: stage 1 locates the server with the least load within the network, and stage 2 decides the best route to get to this server by using the bandwidth and weight of the platform. This algorithm utilized a mathematical example to demonstrate this process. However, this algorithm's key limitation is that it does not have QoS parameters such as cost optimization to quantify network delay.

3.2.2. *Link load balancing*

In multiple paths SDN network, modern approaches have been reported in relevant literature to address high controller LB in multiple path networks. Most of these techniques are meant to optimize path load and choose the least loaded path for the incoming data requests to overcome congestion in the data plane. Moreover, the link LB technique is categorized into three parts: meta-heuristic algorithms, machine learning algorithms, other algorithms.

3.2.2.1. Meta-heuristic algorithms for link load balancing. SDN offers a perfect architecture for network-wide optimization using artificial intelligence (AI) to create an open, scalable, programmable, and manageable network. Also, network-wide management and optimization challenges usually require a vast space for solutions, a large number of variables, and several targets. Heuristic algorithms can solve these problems in a reasonable time but are generally restricted to particular circumstances of the problem (LIAO et al., 2017). Many algorithms have proposed using heuristic algorithms coupled with the SDN network to

improve LB performance.

Hopps et al. (Hopps, 2000) proposed equal-cost multi-path algorithm (ECMP). This algorithm distributes flows across the most available paths based on flow hashing methods. However, the limitation associated with this technique is that of two or more long streams in which conflicting hashes are trying to share the same output port result in bottlenecks in the network. Usually, the flows to paths of the static mapping are not normally specified either for present utilization of the network or size of the flow, thereby leading to collisions which mostly overwhelm the buffer associated with the switches as well as a reduction in the overall utilization of both the link and switch. To overcome the restriction of the ECMP, detection of a huge amount of elephant flows (EFs) is possible by the edge switch or even the terminal host. This allows the central controller to calculate the suitable path while making sure that small flows (mice flows, MFs) become useful on the ECMP switch route forwarding. Nevertheless, this solution can cause the switch or the host to experience a very high bandwidth and processing overhead (Al-Fares et al., 2010; Curtis et al., 2011a; Zhang et al., 2018).

A fuzzy synthetic evaluation mechanism (FSEM) was proposed in (Li et al., 2014) to address path LB. The shortest path selection procedure in FSEM used the Top-K algorithm. In this method, the OpenFlow switch is used to allocate the network traffic to the paths, whereas the central SDN controller is charged with installing the flow-handling rules. The FSEM permits dynamic path adjustment based on the global network view. The researchers implemented their proposed method using the POX controller platform. They carried out comprehensive experimentation by using Mininet simulation to assess the effectiveness, efficiency, and reliability of the implemented method. Experimental findings reveal that the approach successfully adopted the transmitting paths while avoiding abrupt breakdown, which is caused by path failures. However, this LB algorithm may not precisely show the real-time load conditions of the various paths simultaneously since hop counts which may not include all paths are used in the Top-K algorithm.

In a more viable approach, Dobrijevic et al. (2015) proposed an ant colony optimization (ACO) for quality-of-experience (QoE) aware flow routing. The user session parameters are delivered by the application of SDN to the controller in such a way that it triggers the ACO algorithm on a weighted graph, which defines how weights between vertices of each network device are delayed and their loss rate. The corresponding QoE model (i.e., audio, video or data) in relation to the flow type and the estimated value is a factor upon which the fitness function depends. For the shortest path routing approach, the ACO achieved about 24.1% increase for the maximum QoE value. However, this approach depends on the weighted graph, using the weight between vertices as the loss rate and delay for an individual network device. Hence, these parameters are not sufficient to obtain the shortest path.

Lin et al. (2016b) presented a control traffic LB in SDN known as the polynomial-time approximation algorithm (PTAA). The researchers formulated the problem as a framework for nonlinear optimization aimed at finding the optimal number of control traffic accelerating paths for an individual switch and to minimize control traffic delay. Their findings showed that the PTAA method resulted in a minimum of 80% delay reduction in the term of communication efficiency. A link LB algorithm using ACO in SDN was proposed in (Wang et al., 2016a). The experimental result showed this algorithm improved SDNs link LB, network throughput, flow acceptance rate, and effectively reduced packet loss and flow delay in the network. A genetic-ant colony optimization (G-ACO) SDN LB system was introduced in (Xue et al., 2019). The proposed G-ACO SDN LB system combines the GA's mutation, crossover and collection operations with the ACO algorithm for an increased track search speed. A comparison of the proposed scheme with ACO and round-robin algorithms was also carried out, and experimental results show the proposed scheme had better packet transmission and time rates. Thus, the proposed scheme finds the route and the LB was successful.

In a similar work, Jamali et al. (2019) introduced an SDN LB scheme, using a genetic programming LB technique called GPLB. This proposed approach formulates the problem of finding a path as identifying the bottleneck switches as 1) ones with the lowest capacity; 2) that has the shortest path, and 3) with the smallest operations. To select the path with the least load in real-time, GPLB calculates the total load from all the paths using information gotten from the SDN controller. Then, the least loaded path is identified and sent to the controller in response to load information that the LB algorithm receives. The simulative analysis of the GPLB showed a considerable increase in efficiency metrics and a reduction in latency and jitter. The GPLB outperforms similar approaches that have been reported in the literature to perform well in heavy traffic scenarios. The results also showed that the model is better with little or no overhead. In another research, Li et al. (2020) proposed a combination of genetic and ant-colony algorithms (GA-ACO) for dynamic flow scheduling. Under this SDN architecture, the proposed GA-ACO algorithm is used to gain a global view of the network after which it determines and re-routes EFs on the congestion bridge to the globally optimal path. The simulation results showed that, compared to the ECMP and ACO-SDN algorithms, GA-ACO does not only minimize the full usage of links but also significantly increase the bandwidth. This method has some limitation due to the overhead in the EF detection method.

3.2.2.2. Machine learning algorithms for link load balancing. Several studies have proposed the use of machine learning (ML) algorithms coupled with the SDN architecture for improved routing performance (Xie et al., 2018). In SDN, network operators can place some centralized logic on the control plane, which makes it easier to get a global view of the network of ML algorithms.

Chen et al. (Chen-Xiao and Ya-Bin, 2016) presented a solution to balance load distribution that considers real-time path load scenarios. This approach leverages features like the utilization ratio of the bandwidth, rate of the packet loss, transmission hops, and latency used to train a backpropagation artificial neural network (BPANN) to identify load path conditions. The proposed work was implemented in Mininet simulation, and Floodlight controller for determining network efficiency, and experimental results showed improved network performance with a 19.3% achievement in terms of network latency. This technique ignored the types of services that avoid discovering the exact shortest path. An SDN based artificial neural networks (ANN) LB approach was introduced in (Patil, 2018). The proposed method used six features that include packet overhead, latency, hop count, packet loss, trust and bandwidth ratio to improve transmission efficiency. Using this function entails finding the load on every node. The least loaded direction of the transmission is then picked in real-time as a new incoming data flow.

A similar work by Ruelas et al. (Ruelas and Rothenberg, 2018) also presented a knowledge defined networking (KDN) LB system based on a class of the feed-forward neural networks called multilayer perceptron (MLP). The KDN aims to operate and control computer networks using artificial intelligence (AI). KDN advances SDN with what it termed knowledge plane whose features include advanced network analysis and telemetry. The proposed method builds an artificial neural network model of traffic behavior using latency and traffic measurements across several tasks to predict network performance. The ANN model is trained to select the route with the least load. Experimental results point to a possible performance gain in terms of efficiency while using KDN.

Moreover, Hou et al. (2019) proposed a new LB algorithm called a maximum probability fitting (MPF) by using a multinomial logit model (MNL). MPF selects probabilistic rerouting for every flow, and unlikely to distribute the traffic to the same link. Findings revealed that MPF achieved a 3.6% increase in throughput and a 26.84% decrease in the rate of packet loss compared to offline increase first fit (OFF).

Rupani et al. (2020) introduced an LB approach by using the global network view property in SDN. The study aimed to determine the best

route for data transmission and achieve a significant reduction in network latency. To achieve this, several relevant features were collected from each route. The load balancer measures the throughput, the rate of packet loss, latency, the number of hops and the usage of nodes. Using these features, the overall load condition is predicted for various shortest paths given by Dijkstra's algorithm via a trained neural network model. Dijkstra's algorithm uses the link's transfer rate as a metric. In real-time, the ANN is evaluated to make full use of the relation and node. As the total node utilization along each path is also extracted and fed to the neural network, the highest node utilization is achieved with the least load. The load balancer sends the path details to the SDN controller after selecting the route with the least load. Afterwards, the SDN controller moves the flow rules in the switches along the best direction provided by the load balancer. If a link or node fails along the best path, then the load balancer selects the second-best path in real-time in order to complete the data transfer with minimal delay. Thus, this system achieves optimum overall network utilization in real-time. Additionally, this approach ignores the types of services that avoid discovering the exact shortest path.

In another research, Zhang et al. (2020) introduced a critical flow rerouting reinforcement learning scheme (CFR-RL). This scheme uses reinforcement learning to automatically learn a critical flow selection strategy without any domain-specific rule-based heuristic approach. Critical flows are selected by CFR-RL for each traffic matrix and rerouted to utilize the network link optimally. By solving the rerouting optimization problem. The evaluation results showed that CFR-RL achieves close to optimal efficiency by simply rerouting a small subset of the entire traffic. Also, CFR-RL generalizes well to traffic matrices, although it was not included in the training set. Yu et al. (2018) proposed an ML-based SDN architecture using the deep deterministic policy gradient (DDPG) with a deep reinforcement learning method to automate the SDN routing process. This approach also suggested a DDPG routing optimization system called DROM to realize the control and management of the regional, real-time, and personalized network information in continuous time. Experimental results have shown that DROM is durable consistent and highly productive, DROM is also capable of boosting network performance with reliable and superior routing services compared to existing routing solutions.

Moreover, both (Sun et al., 2019b, 2020b) proposed an intelligent yet scalable network control (SINET) architecture to optimize routing. To increase robustness and scalability, SINET selects multiple important routing nodes to be directly managed by a deep reinforcement learning (DRL) agent, which dynamically creates routing policies to maximize network performance. The results of the simulation revealed that SINET could reduce the average completion time for a network with 82 nodes by at least 32% and display better robustness against minor topology changes than other DRL-based schemes.

3.2.2.3. Other algorithms for link load balancing. Load balancing method in the data plane is used to improve network performance and enhance its ability to meet QoS needs. This approach compared three different controller types, spanning tree protocol (STP) based controller, LB-based controller, and hub-like controller, for a while evaluating the impact of LB on given network performance indicators. The experimental findings showed a reduction in network congestion while the network efficiency generally increased (Chahlaoui et al., 2019).

Silva et al. (da Silva et al., 2020b) proposed a dynamic LB algorithm for traffic on the data plane that minimizes the impact of heavy data traffic on the network while mitigating bottlenecks. This algorithm dynamically alters the flow of ties as the network use intensifies. The algorithm identifies the shortest paths and calculates the cost of connection by selecting the best link after detection and type of bottlenecks. This reduces the loss of latency and packets in network congestion scenarios. Experiments were performed using the Mininet simulation and OpenDaylight Controller. By using emulation testing, a major improvement was observed in jitter, packet loss, and throughput. The proposed algorithm can function in any form of SDN setting, such as balancing the flow of traffic on the data plane.

On the other hand, a dynamic scheduling scheme called AggreFlow was proposed in (Guo et al., 2016). AggreFlow schedules flow in a coarse-grained operate-set fashion, employs lazy rerouting to amortize the enormous number of simultaneous rerouting operations over a relatively long time, and adaptively redirects operate-sets to maintain load balancing on active routes. Simulation results showed that AggreFlow achieves high power efficiency and excellent, low overhead LB performance. Zeng et al. (2019) through the use of link bandwidth algorithm (LBLB) presented an effective LB based on a data center that uses the bottleneck and the average of the remaining bandwidth links in the route to avoid heavy links. This technique showed improvement in LB performance as well as reduction of link congestion and packet loss. A tabular presentation of the compared methods is illustrated in Tables 2 and 3.

3.3. Load balancing for control plane (LB for CP)

CCControl plane LB balances loads over distributed controllers to avoid bottlenecks associated with huge SDN network within a centralized controller. Based on studies that have been conducted on multiple controllers, it can be categorized into logically-centralized/physically-distributed, distributed LB, which is divided into hierarchical, and horizontal, and virtualization controller LBs.

3.3.1. Logically-centralized/physically-distributed controller load balancing (Log cent/phy dist LB)

This kind of LB is known as a logically-centralized LB and physically-distributed controller, Fig. 3 describes the structure of the logically C/P distributed controller LB. The controllers are centralized logically and physically in a distributed controller architecture. Each controller can only communicate or decide for the domain for which it is responsible. It differs from the logically centralized design where every controller views the global network to make a decision (Blial et al., 2016; Tadros et al., 2018).

Although logically-centralized/physically-distributed LB is founded on the principles of multiple controller designs, it assumes there is a single logical controller. Logically-centralized/physically-distributed architectures were proposed to address the use of replicated controllers and solve the SPOF problem. The previous ideas are disadvantageous in that inactive controllers will not be activated until the main controller fails (Nunes et al., 2014). A similar system is presented by HyperFlow (Tootoonchian and Ganjali, 2010) with a distributed control platform as an application of the NOX controller (Gude et al., 2008). It can be used to carry out state distribution that exists among distributed controllers by applying a subscription system (called WheelFS) within the distributed file system (Stribling et al., 2009). Nevertheless, the HyperFlow can only handle a few thousand events per second, which is a scalability limitation. The sequence assignment algorithm called SeqAsn for handling regular flow variations between the flow paths and the controllers was proposed by (Guo et al., 2020b). The proposed method sought to explore the possibility of saving controller resources using switch assignments. This approach studied the flow requirements of controller resources on distinct paths in a distributed control plane and discovered the possibility of resource-saving via switch assignments. Thus, the problem of switch assignment is formulated as an integer program to reduce controller resources requirement. Results from the simulation showed that SeqAsn algorithm substantially saves controller resources as opposed to current solutions.

A technique called load variance-based synchronization (LVS) only performs valid state synchronization between the controllers, especially when a certain threshold of a server or domain is exceeded. Consequently, LVS can proficiently decrease the synchronization overhead between controllers and, thus, eliminate the problem that often arises in

Table 2
Selected comparison between various data plane load balancing solutions in SDN.

Year [Ref]	Controller	Types of Algorithm	Simulation/ Real Environment	Types of data plane LB					Remarks
				Server LB		Link LB			
				Static LB	Dynamic LB	Meta. LB	ML LB	Other LB	
2010 (Uppal and Brandon, 2010)	NOX	Load-based, random, round-robin	Real environment	✓					• Provide LB services depending on the current lowest load on servers. • This study allows only one controller to manage a switch. • A single point of failure (SPOF).
2014 (Li et al., 2014)	POX	FSEM, Top-k paths	Mininet			✓			• Proposed FSEM to select the shortest Top-k paths, using the top-k algorithm. • The Top-k algorithm considers only the hop count that likely leads to a path great enough to be removed.
2015 (Dobrijevic et al., 2015)	Open Daylight	ACO	Mininet			✓			• For QoE-aware flow routing, it provides ACO approach. • The weight between the vertices depends on each network device delay and loss rate parameter, which is inadequate.
2016 (Chen-Xiao and Ya-Bin, 2016)	Floodlight	BPANN	Mininet				✓		• This research used BPANN algorithm. • This method depends on some features such as bandwidth utilization ratio, packet loss rate, transmission hops and transmission latency.
2015 (Kaur et al., 2015)	POX	Load-based, random, round-robin	Mininet	✓					• Proposed an algorithm for load balancing based on SDN.
2013 (Shang et al., 2013)	Floodlight	SBLB	Mininet		✓				• Proposed dynamic LB. • Neglected types of services.
2018 (Patil, 2018)	-	ANN	NS 2				✓		• This study used BPANN algorithm. • This method depends on some features such as hop count, bandwidth ratio, packet overhead, latency, trust and packet loss.
2018 (Ruelas and Rothenberg, 2018)	Open Daylight	KDN based on MLP	Mininet				✓		• This research used KDN based on MLP a class of feed-forward ANN algorithm. • This approach can predict network efficiency given traffic parameters through the creation of a traffic behavior model that utilizes latency and bandwidth measurements across different tracks.
2018 (Abdelltif et al., 2018)	Floodlight	SBLB	Mininet		✓				• This study takes into consideration the type of service request to optimize server usage and minimize user response time.
2019 (Jamali et al., 2019)	Open Daylight	GPLB	Mininet			✓			• This study formulates the problem as a mathematical model in SDN by taking into account the limited size of the switches while maximizing the network link usage. Calculating the amount of the use of links in a route called route cost • This approach aims to explore the trade-off between the cost of the route and the cost of service.
2020 (Srivastava and Pandey, 2020)	–	LB based on channel capacity	Mininet		✓				• This research focused on the use of the channel capacity over the server load to dynamically determine the best route.
2018 (Yu et al., 2018)	–	DDPG with DROM	OMNeT++				✓		• This research proposed an SDN DROM method for achieving uniform and customizable optimization of the path.
2016 (Guo et al., 2016)	–	AggreFlow schedule	NS-3					✓	• This research suggested scheduling of dynamic flows that achieved power efficiency in DCNs and improved QoS.
2019 (da Silva et al., 2020b)	Open DayLight	Dynamic LB	Mininet					✓	• This technique identifies the shortest paths and calculates the cost of connecting by selecting the best connection after finding and type of bottlenecks. • This technique uses information from network equipment to locate bottlenecks in the data plane via the controller and to redefine the packet route where there is significant traffic flow, taking into account the best path at the time of transmission.
2019 (Xue et al., 2019)	Open DayLight	G-ACO	Mininet			✓			• The proposed G-ACO approach combines GA's collection, crossover, and mutation process

(*continued on next page*)

Table 2 (*continued*)

Year [Ref]	Controller	Types of Algorithm	Simulation/ Real Environment	Types of data plane LB					Remarks
				Server LB		Link LB			
				Static LB	Dynamic LB	Meta. LB	ML LB	Other LB	
									with the ACO algorithm to improve the speed of path discovery and the ability to search for an optimal path.
2020 (Sun et al., 2020b)	–	SINET	OMNet++				✓		• This work proposed a DRL-based dynamic routing scheme to adjust real-time traffic routing policy dynamically.
2020 (Zhang et al., 2020)	–	CFR-RL	TensorFlow				✓		• This study introduced a CFR-RL scheme that performs critical flow selection accompanied by linear programming rerouting.

Table 3
Selected comparison between various data plane load balancing solutions in SDN (Continued).

Year [Ref]	Controller	Types of Algorithm	Simulation/Real Environment	Types of data plane LB					Remarks
				Server LB		Link LB			
				Static LB	Dynamic LB	Meta. LB	ML LB	Other LB	
2015 (Chen et al., 2015a)	Floodlight	Round-robin, random, SBLB	Mininet		✓				• This method cannot be used for real-time load monitoring. •Used reactive flow entry.
2015 (Du and Zhuang, 2015)	Floodlight	Dynamic server LB	Mininet		✓				•This method used sFlow provided by flow statistics counter.
2017 (Zhong et al., 2017a)	Floodlight	LBBSR	Mininet		✓				•This approach did not consider energy savings in server LB.
2016 (WANG et al., 2016)	Floodlight	Dynamic LB	Mininet		✓				•Proposed a dynamic load balance algorithm in a cloud center based on SDN to also improve throughput. • This approach did not consider the latency and utilization metrics.
2017 (Wang et al., 2016a)	Floodlight	ACO	Mininet			✓			•Provide an LB algorithm using ACO in SDN. •Improving the SDNs link LB, network throughput and flow acceptance rate, and effectively reduce the network flow delay and packet loss rate.
2019 (Hou et al., 2019)	ONOS	MPF	Mininet				✓		•This method used probability to select a rerouting path for each flow and is reduces the likelihood of distributing traffic to the same link.
2019 (Zeng et al., 2019)	–	LBLB	Mininet					✓	•This method proposed an LBLB mechanism that is used for both package-in and rescheduling phases.
2019 (Chahlaoui et al., 2019)	POX	LB for data plane	Mininet					✓	•This study provided a comparison in terms of jitter and delay among different controllers such as a hub-like controller, an augmented controller with STP, and an LB controller.
2020 (Rupani et al., 2020)	Floodlight	BPANN	Mininet				✓		•This research used BPANN algorithm. •This method depends on some features such as packet loss rate, bandwidth utilization ratio, transmission hops, transmission latency, and total node utilization.
2020 (Li et al., 2020)	Floodlight	GA-ACO	Mininet			✓			•The GA-ACO algorithm makes use of ECMP to schedule all incoming data flow initially. • If the link utilization ratio reaches a certain threshold, GA-ACO is called upon to determine the best route to reroute EF on the congested connection.

a forwarding loop. On the other hand, the distributed OpenFlow controller focuses on reducing the synchronization overhead between controllers in different domains without implementing an LB solution (Guo et al., 2014). In SDN based mission-critical networks LB scheme, Hai et al. (Hai and Kim, 2016) proposed a distributed controller. The scheme could ensure a balance in the control plane to achieve a high resource usage, reliability, and resilience within the network, which are characterized by the load status and the coefficient of the dynamic weight. Besides the use of the proposed technique predefined load threshold, communication overhead was significantly reduced when compared to the bench-marked algorithms. The efficiency of the priority queuing module technique is also demonstrated especially for the network that is grouped into two categories of traffic; the critical traffic such as in-line traffic (VoIP or VoD) and non-critical traffic like offline traffic (HTTP or FTP). In order to ensure better QoS, the priority procedure ensures that the controllers process the critical traffic before the non-critical ones based on the necessary information in the packet header.

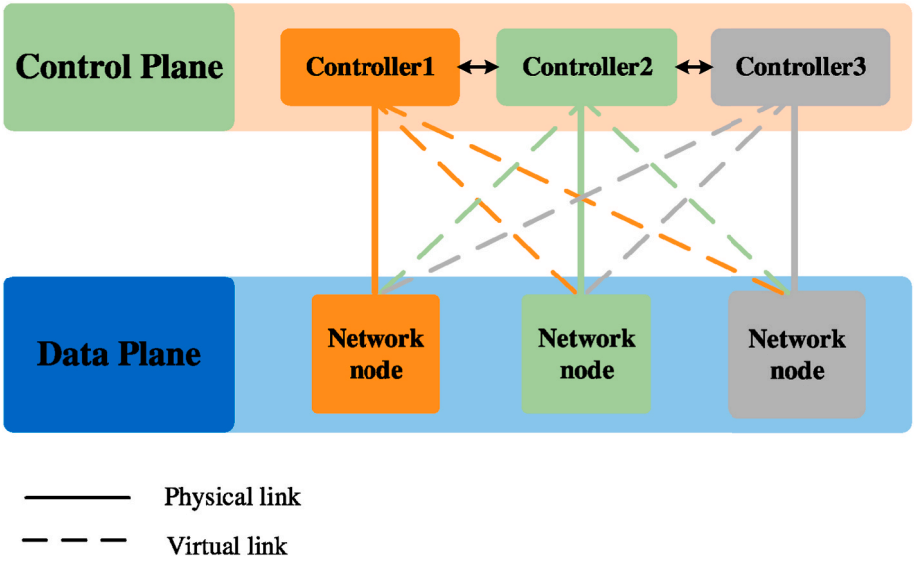

Fig. 3. The logically-centralized/physically-distributed controller load balancing.

A smart cooperative LB and security framework for SDN distributed controllers, called SCPLBS was proposed in (Zhong et al., 2019). The proposed technique refers to a single domain network architecture for distributed controllers with SDN. This method employed the Floodlight controller to build the restlet-based cooperative framework. SCPLBS includes a module each for message authentication, data collection, LB, and a module for recovery of failures. Experimental outcome and analysis showed that the platform realizes the LB and recovers from the failures of the distributed controller based on safe communication. However, this technique is disadvantageous in changing the controller's position over the OpenFlow switch, which can result in a brief delay of the primary business.

A new control panel model has been proposed in (Sahoo et al., 2019a), in which demands for control decisions are treated separately. In accordance with the architecture of the control plane, the load in the local area network (LAN) are predicted by L2 controllers predict for L3 controllers in the wide-area network (WAN). This alertness-based load prediction approach has been proposed to reduce the burden on the controller. While this approach considerably reduces the flow's timeout value, additional overhead in delay often results from the flow entry's initial packets, which leads to a further error of prediction. However, the proposed method reduces the flow's timeout value. Furthermore, attention has been paid to reducing the response time between the controller and the router. Extensive simulation has proven the feasibility of the proposed scheme.

3.3.2. *Distributed controller load balancing*

Distributed controller LB architectures entail using one or more controllers in networks to address challenges faced by a single controller network. The rationale for this is to ensure that controllers that can permit the sharing of load equally in the network are formed. Besides, one controller can take over from another controller should a crash occur. Most of these sophisticated procedures require limited technological implementations from distributed systems. It has been reported in pertinent literature that distributed controller architectures are not always based on the multiple controllers (Eko OktianSangGon Lee, 2017; Chahlaoui and Dahmouni, 2020). This type of network a physically distributed and different based on the location of the controllers and the communication type (Blial et al., 2016).

The main problem with distributed controller solutions is related to retaining consistency within various distributed controllers. In addition, various applications in the network (Yang et al., 2014) tend to be given minimal consideration by the distributed controllers due to the inconsistency that exists among the controllers as seen from the global view of the network (Azodolmolky, 2013). Moreover, multiple controller scenarios raise the need for resource management such as the distribution of controller state, consistency and sharing of data in addition to extended propagation delay, which affects the response time to network events such as packet-in messages (Abdelaziz et al., 2017).

LB between multiple controllers is another issue that must be considered. LB among several controllers is of two types, namely; centralized decision and distributed decision. Fig. 4 illustrates a centralized decision (Liang et al., 2014; Dixit et al., 2013) where there is a coordinator controller which plays the role of a super controller that responsible for ensuring better maintenance of a global controllers load info table. There are two essential processes which are involved; a collection of land information of all controllers and ensuring that the LB command is sent to the overloaded controller. The two processes can prolong the time of the LB. Conversely, controllers with the distributed decision (Zhou et al., 2014b; Zou et al., 2017; Zhong et al., 2019) tend to exhibit similar features in the SDN network when they are connected.

In contrast to centralized decisions, distributed decisions have preferable performance that is suitable in terms of responsiveness, reliability, and scalability as a result of their physical distribution and logical centralization. On the other hand, distributed controller LB architectures are divided to organize the controller's LB as hierarchical and horizontal LB.

3.3.2.1. Hierarchical controller load balancing. In the hierarchical controller, each controller is located at different layers; Fig. 5 shows the structure of the hierarchical controller. Kandoo in (Hassas Yeganeh and Ganjali, 2012) allocated controller status by replacing such a controller with that of a two-level hierarchy that has a root and numerous local controllers. However, it has become impossible for the controllers

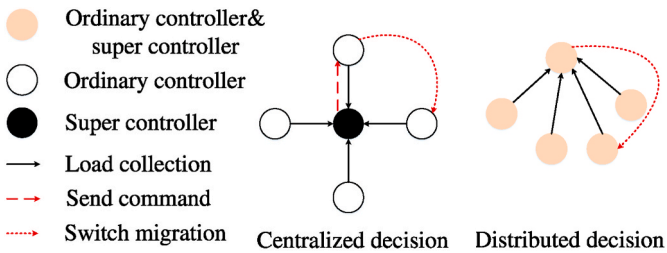

Fig. 4. Centralized decision and distributed decision (Zhou et al., 2014b).

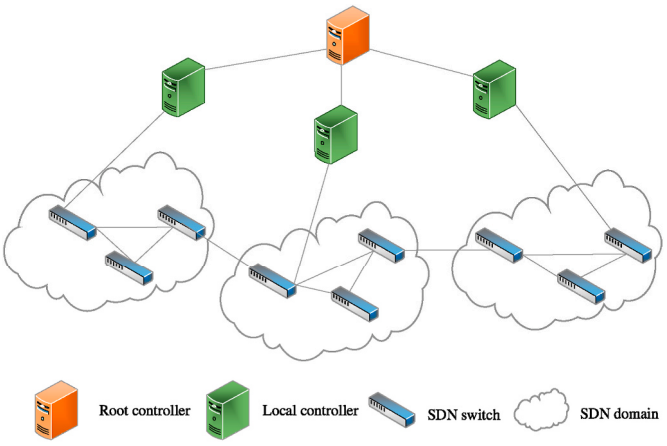

Fig. 5. The hierarchical controller.

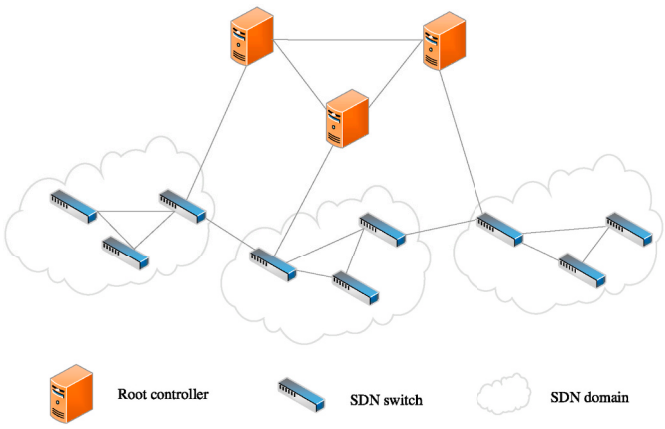

Fig. 6. The horizontal controller.

within the tiers to communicate with each other within the system, and thus, it limits the utilization of the second-tier services which is required for a global view of the network.

In a multiple-controller SDN network, an LB mechanism was presented in (Ma et al., 2017). This technique introduced a meta-controller (MC) that is based on a manager with several controllers that can be used for control in the SDN in a decentralized manner. The MC-based manager mechanism was established to progress the performance of several controllers and ensure the successful loading of each controller. Therefore, this technique introduced a scheduling plane based on local control that can be used to discover the best way to distribute the available local controllers. The local control plane based scheduling solution builds up its information base through the MC-based manager, including the controller statuses, network traffic, and the CPU. To provide an efficient solution in a distributed controller environment, a hierarchical control structure should be applied. Eventually, the enhancement of logically centralized control becomes possible due to the single controller architecture's scalability and efficiency as shown from the use and study of SDN controller resources. This technique constructs an optimal processing performance network. Findings have shown that MC based management technique reduced the control plane loading by increasing the bandwidth utilization in SDN. However, this approach ignores the type of service required to implement the application LB model for optimal resource usage in SDN environments.

In multi-layer hierarchical SDN, Lin et al. (2016a) introduced a reinforcement learning-based QoS-aware adaptive routing (QAR) technique. It first introduced the distributed hierarchical control plane that minimizes signaling delay, which serves as realistic SDN architecture. Their aim of introducing the QAR was to enable time-efficient and addictiveness upon the proposed architecture situated on the QoS-aware packet forwarding. The result revealed that in terms of QoS provisioning, the QAR technique outperforms the conventional learning approach with quick convergence. Likewise, this scheme enables, in practical, large scale on-line, QoS-aware routing on the network implementation of the SDN and the defined software services. Laman is a framework that provides scalability using a supervisor controller (Nayyer et al., 2019b). This method aims to retain a centralized architecture approach and enhance the SDN scalability. In the SDN network, the control is distributed between the supervisor controller and application-specific controllers. The supervisor controller is a central controller that links all the other controllers. Laman framework reduces the load from the central controller by distributingthe main incoming load to enhance scalability. Evaluations results showed that Laman frameworks provide higher performance than similar existing frameworks.

3.3.2.2. Horizontal controller load balancing. All controllers are positioned on the same layer in the horizontal controller LB as illustrated in Fig. 6. Thus, it is also referred to as a flat controller. Although each controller founds diverse domains, all the controllers are viewed as root controllers which are familiar with the entire state of the network (Zhang et al., 2017).

Through east-west interfaces, there is communication between each controller and the management of the network state is carried out by the controllers with equal state (Benamrane et al., 2017; Lam et al., 2016). In this approach, every single controller controls a specific domain. There are two strategies for the implementation of distributed architecture; the local and global view strategies. In the local view approach, logical nodes are an abstraction of neighboring networks, whereas the global view strategy is for the entire network. However, communication between controllers happens through specialized channels to share network information (e.g., information on reachability) about their domains (Karakus and Durresi, 2017b). A link LB mechanism was proposed in (Shang et al., 2018) for SDN based on flat controllers known as the service-aware adaptive. This method was intended to overcome the imbalance in the network load and scalability challenges in the control plane. Results showed that this method increased the average connection bandwidth utilization rate to 79%, as well as jitter load and average link latency. Besides, it guaranteed the requirements for the QoS operation.

A similar work by, Canini et al. (2013) proposed a software transactional networking (STN) along with middleware in a distributed controller to address policy inconsistency in diverse distributed controllers in the data plane. While it is expected that ONIX application developers provide needed logic to discover and overcome network conflicts resulting from concurrent control, the proposed approach introduced a composition of concurrent policy mechanisms such that in a general fashion, it can be used by any application. The possibility of adapting multiple controllers to increase the scalability without the incorporation of the global network view and storing information about the network topology in SDN controllers deployed in the data center has been studied in (Tam et al., 2011). The researchers introduced two heuristic methods, namely path-partition and partition-path, by using flow routing to examine the practicability of controllers.

A cluster-based approach for distributed controllers was proposed to maintain the inter-communications between controllers yielding more scalability and flexibility in (Yazici et al., 2014). Their developed architecture, being cluster-based, offers the network elasticity regardless of addition or removal of controllers because it can be featured as the network application provider. A master controller is then selected from amongst the available controllers to be in charge of detailing both the switches and controllers. This developed distributed setup of controllers reduces the scalability of SDN networks. Moreover, they are not always suitable to provide the planned scalability due to an unbalanced load that is seen across the controllers resulting from the number of switches

being determined by network administrators. Consequently, this may cause controller overload.

Onix is one of the distributed controllers applied for large-scale network architectures that enable multiple SDN controllers (Koponen et al., 2010). It is used to handle the collection and dissemination of information of various kinds from the switch. This also assigns suitable controls between several controllers. Dixit et al. (2014) introduced a method to distribute the load among the controllers via controller pool evenly. It works by elastically incorporating or nullifying the controllers via the controller pool by the load and predefined threshold values. On the other hand, Phemius et al. (2014) proposed a framework called distributed SDN control plane (DISCO) to share aggregated wide network information for controller view consistency. It introduced two parts, namely the intra-domain and inter-domain. The intra-domain is in charge of the controller's main domain. At the same time, the inter-domain is responsible for flow management through distributed networks such as multi-protocol label switching (MPLS) tunnels. Bari et al. (2013) discussed the problems of integrating distributed controllers in a wider area network and developed a mechanism that dynamically alters the necessary active controllers with switch numbers to minimize flow setup times, vertical and horizontal overheads.

A bidirectional matching strategy (BMS) was proposed in (Hu et al., 2018b). The algorithm was designed to implement multi-controller deployment. First, the matching lists of switches and controllers are built by collecting hops, delays, and traffic in the network. A mutual selection strategy is designed by the sequence of the elements in a matching list from which the best elements are selected by the switch and controller to execute the correspondence in turn before the distributed network is constructed. The simulation showed that BMS could achieve better implementation of multi-controllers compared to existing algorithms. Moreover, a dynamic controller workload balancing scheme, named MARVEL was proposed in (Sun et al., 2020a) based on multi-agent reinforcement learning for switch migration actions generation. MARVEL operates in offline and online modes for preparation and decision-making, respectively. Each agent learns via a training process to move switches based on their interaction with the network. MARVEL is used to make decisions on migratory switches in online mode. Experimental findings showed that MARVEL outperforms existing competitive schemes by improved processing of the control plane's request by a minimum of 27.3% with minimal time; about 25% lesser processing time than competing approaches.

3.3.3. *Virtualization controller load balancing*

Another type of control LB using slices technique is based on SDN network virtualization. In this approach, the physical network infrastructure has a virtual layer is placed above it. By controlling packet routing and load balance, a virtualized network can be achieved by stimulating idle wires to share their load (Fratczak et al., 2013; Chen et al., 2016). Network virtualization is provided in this layer through the construction of virtual networks, which are made up of virtual resources like routers, switches, and other nodes that are to be managed and controlled. A transparent proxy which links multiple controllers on one of the networks to a side of the switch is used to implement control (Kashiri et al., 2016).

An OpenFlow virtualization controller, called FlowVisor (FV), which serves as a transparent proxy between the switches of the OpenFlow and that of the several OpenFlow controllers was proposed in (Sherwood et al., 2009). With FV, multiple isolated virtual logical networks (slices) can be created with various addressing and flow forwarding methods on one physical infrastructure such that diverse controllers in diverse slices can share similar network resources (e.g., OpenFlow switches/ports). Slices can be defined in layer 1–4 by any combination. The layer model is the same as the definition in a network. When sliced by switch ports, the policy is implemented in layer 1. When the Ethernet address or type is specified, slicing is implemented in layer 2, while enforcement in layer 3 is carried out by the src/dst IP address or Type. Layer 4 makes use of the src/dst, TCP/UDP port or ICMP code/type. Within each slice, the FV is responsible for imposing an isolation policy to avoid the control of packet traffic of one slice from other slices (Qian et al., 2017). The FV intercepts OpenFlow messages from guest controllers, as illustrated in Fig. 7. (1) Use a slicing strategy for the recipient, (2) rewrites the message transparently, (3) controls only one network slice. (4) Only on compliance with their slice policy, will messages from switches be forwarded to guests (Köhler, 2018; Sherwood et al., 2010a).

FV and OpenVirteX (OVX) (Al-Shabibi et al., 2014) are two examples of proxy controllers that are used to create slices for virtual networks. LB architectures with multiple services that utilize slicing techniques aim to use each of the multiple controllers that are also connected to the FV controller for each network service to provide LB strategies. A slicing mechanism is utilized in this technique to manage various controllers at different parts of the network. The FV (Sherwood et al., 2009) slicing mechanism depends on the packet header's field information, which determines where the packet is forwarded. For instance, fresh requests for destination port 80 are forwarded to the HTTP LB controller, whereas requests for destination port 21 are forwarded to the FTP controller LB. These requests first pass through the FV, which manages all network slices and all related services. Flow entry is inserted by FV within the switch in line with the packet header's information and transmitted by the incoming packet to the corresponding controller as developed in (Koerner and Kao, 2012).

One of the virtualized schemes is the network slicing technology that helps network managers in the virtualization process of network resources like the topology, bandwidth, and traffic. Chen et al. (2016) introduced the enterprise-visor engine that warrants less resource while adjusting the maximum resource limit according to current status. Compared with the OpenFlow and FV frameworks, this method enhanced the network's utilization by 25.7% and 13.4%, respectively. Moreover, Jin et al. (2019) proposed TALON, a traffic LB system for the distribution of the throughput. TALON assigns a flow capacity per tenant by calculating multiple routes to satisfy the traffic requirement. The design and implementation of TALON used an open-source hypervisor network. The evaluation results showed that the standards are almost met for the throughput of each tenant. Besides, the throughput increases by up to 2.29 times compared to a non-LB network. A comparison of the solutions above is presented in Table 4.

3.4. *Other aspects of data/control planes load balancing (Other aspects of D/CP LB)*

An overview of the other aspects of data/control planes LB like SDN switch flow table LB, control plane LB during controller failures, switch migration mechanisms, management SDN controller load, resource

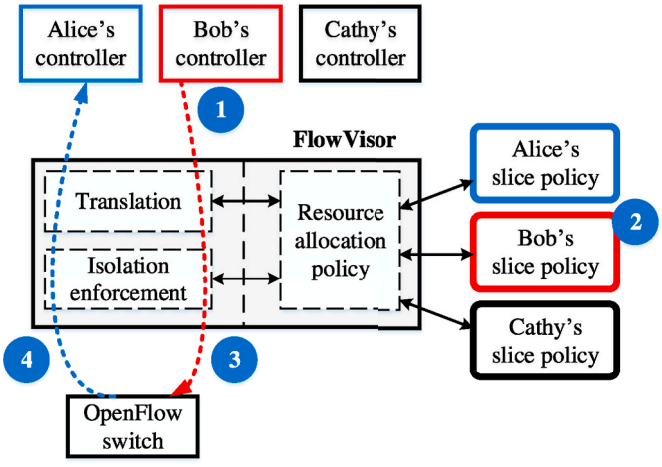

Fig. 7. Flowvisor slices adaption from (Sherwood et al., 2010a).

Table 4
Selected comparison between various control plane load balancing solutions in SDN.

Year [Ref]	Controller	Types of Algorithm	Simulation/Real Environment	Types of Control Plane LB				Remarks
				Log. L B	Distributed Controller LB		Virt. L B	
					Hier. L B	Hori. L B		
2014 (Guo et al., 2014)	POX	LVS	Mininet	✓				•This method focuses on reducing the synchronization overhead among controllers in different domains without implementing an LB solution.
2016 (Hai and Kim, 2016)	Floodlight	Dynamic and adaptive LB	Mininet	✓				•This method has some limitation in the analysis of the types of traffic as it depends on only weight coefficients which are not enough as it needs more statistical analysis of the types of traffic.
2017 (Ma et al., 2017)	NOX	LB mechanism based on MC	Real environment, Mininet		✓			•The types of services needed to achieve an application LB model for optimal usage of resources in the SDN environment is neglected by this method. •Investigation of energy consumption was not made.
2016 (Chen et al., 2016)	Open Daylight	Dynamic resource management	Mininet				✓	•This method can adjust the maximum resource limit according to its current status and provide minimum resource guarantees for enterprise visor engines.
2018 (Shang et al., 2018)	Floodlight	Service-aware adaptive link LB	Mininet			✓		•This approach examines the benefits of the SDN network to address LB, guarantee QoS and ensure intentional multi-controller deployment and proposes a service-oriented LB system in the SDN.
2010 (Tootoonchian and Ganjali, 2010)	NOX	HyperFlow	Real environment	✓				•This method designed and implemented HyperFlow that enables OpenFlow deployment in mission-critical networks. It comprises of enterprise network associated with the data center.
2020 (Guo et al., 2020b)	–	SeqAsn	Real environment	✓				•This research proposed an allocation problem for the flow path-aware controller. •This research offers a greedy assignment algorithm for adaptation to the regular fluctuation of the flow.
2018 (Hu et al., 2018b)	Open Daylight	BMS	Java, Matlab			✓		•This research suggested the multi-controller deployment process and transformed the flow transmission into a queuing model. • This work considers delay and traffic are two major factors influencing the multiple controller deployment.
2019 (Nayyer et al., 2019b)	Ryu	Laman	Mininet		✓			• This method proposed an effective hierarchical scalable SDN framework. •Minimizes central controller supervisor load by attaching Contributory SDN controllers with it. •Enhances the use of the SDN central controller for CPU, flow setup and packet throughput.
2019 (Jin et al., 2019)	ONOS	TALON	Mininet based on OVX				✓	•TALON has key functionality in the allocation of the throughput. It calculates the paths of packet forwarding for the specific requirement of throughput. •TALON architecture is designed to split multi-paths by adding flow rules to realize multi-paths calculated in the physical network.
2019 (Zhong et al., 2019)	Floodlight	SCPLBS	Mininet	✓				•SCPLBS and controllers use a communication method that is secure by the message authentication code. •The collaborative framework utilizes a data collection algorithm which adapts to variation in data to obtain the status of controller and load controller information.
2016 (Lin et al., 2016a)	–	QoS based on QAR	Real environment		✓			•This approach was developed to provide on-line, QoS-aware routing in realistic large-scale SDNs and software service-defined network implementation.
2014 (Yazici et al., 2014)	Beacon	Distributed OpenFlow controller	Real environment			✓		•This approach offers a distributed system with OpenFlow controllers and a related collaboration structure. •This framework supports the dynamic controller removal and addition and removal of controllers into the cluster without little or no interruption in the operation of the network.
2014 (Dixit et al., 2014)	Floodlight	Elastic distributed controller architecture	Real environment, Mininet			✓		•This approach suggested ElastiCon, an architecture for elastic distributed controllers. •Designed and implemented algorithms for switch migration.
2019 (Sahoo et al., 2019a)	Floodlight	Flow entry timeout mechanism	Mininet	✓				• This work suggested a load prediction method that downgrades the QoS parameters for the initial flow packets. • This work focuses on a new architecture for control planes where the load for L3 controllers are predicted

(*continued on next page*)

Table 4 (*continued*)

Year [Ref]	Controller	Types of Algorithm	Simulation/Real Environment	Types of Control Plane LB				Remarks
				Log. L B	Distributed Controller LB		Virt. L B	
					Hier. L B	Hori. L B		
								by, the L2 controllers. predict the load for L3 controllers.
2020 (Sun et al., 2020a)	–	MARVEL	TensorFlow			✓		•This work introduced a distributed control plane model as a multi-agent structure designed to approach the SMP in a distributed fashion. • This method developed a framework for DRL for each agent in the MARL model.

allocation and management in SDN, distributed controller placement, and security issues in SDN LB is presented in the section.

3.4.1. SDN switch flow table load balancing

Being an emerging technology, SDN ensures flexibility in the control of network flow. Unlike the conventional IP network, where routing tasks are performed using traditional routing schemes (like ECMP and OSPF) with unsupervised forwarding devices (like routers and switches) spread across the network, a centralized controller with the capability of globally viewing the network is used in SDN for routing flows and adjusted according to variation in-network status (McKeown et al., 2008b; Al-Fares et al., 2010; Curtis et al., 2011b; Agarwal et al., 2013; Guo et al., 2014). Basic functions of the network (such as forwarding in layer 2, routing in layer 3 and classification of packets in layer-2-4 (Xu et al., 2013; Xu et al., 2009)) are treated/assumed to be flow entries in OpenFlow specifications (Pfaff et al., 2012). However, due to the high power consumption and cost of chips, there is limited flow table space in most majority of switches that are compatible with OpenFlow; hence, as the number of concurrent flows gets large in the network, the flow tables inadequate (Qazi et al., 2013; Zarek et al., 2012). This inadequacy increases the likelihood of overflow of the flow-table. To address this shortcoming, flow entries that are being used are often removed to create space for new flows. Moreover, the flow-table overflow implies that the controller must intermittently update the flow table; resulting in the degradation of the TCP output due to interruption of current flows (Guo et al., 2018).

JumpFlow, introduced in (Guo et al., 2015), uses data from the packet header information field to carry out forwarding/routing tasks. As such, reduces utilization of flow table in SDN while ensuring correct and reactive placement of flow entries in switches. The simulation results showed that JumpFlow reduces the percentage of flow-rejection, prolongs the time of occurrence of first flow-rejection, and increases the ratio of agreed multicast groups to baseline schemes. Furthermore, a novel routing scheme in which optimization of network flow was carried out for effective routing and optimal usage of the limited flow tables space in SDN switches (Jia et al., 2016). The NP-hard problem was solved using the K-similar greedy tree (KSGT) algorithm. Experimental results showed that 60% of flow inputs were reduced using KSGT with about 25% increase in effective forwarding and deployment flows within the same flow table space compared to existing solutions.

Similar work by, Jia et al. (2017) also formulated flow table LB as an optimization problem to reduce and balance entries in the flow table such that it holds extra flows with minimal overheads. The KSGT algorithm performed best in actual network setup on this NP-hard problem despite the associated network MPLS labels overhead and limited space of the flow table in SDN switches. On the other hand, an online routing scheme, called software-defined adaptive routing (STAR), that effectively uses restricted flow-table resources for optimal network efficiency was proposed in (Guo et al., 2017). In particular, STAR detects the use of each switch in real-time flow-table, deletes expired flow inputs intelligently when required to accept new ones, and chooses the paths new flows entries are forwarded according to network-wide flow-table switch usage. Experimental results on the Spanish backbone network showed superior performance of STAR over existing schemes. STAR achieved an 87% reduction in controller workload in new flow routing, 49% reduction was achieved in packet latency while the average flow rate was increased by 123% despite scarce flow table resources.

Guo et al. (2018) introduced a dynamic routing scheme called DIFF. This scheme distinguishes according to their impact on network resources and adapts routing paths to reduce flow-table overload problems and inefficient allocation of bandwidth. A set of paths are pre-generated by DIFF for any pair of edge switches on the source-destination link. To balance flow-table utilization, new flows paths are intelligently selected from the pre-generated path-sets. EFs are adaptively rerouted to reach optimum throughput using the law of allocating max-min equal bandwidth. Experimental results showed the capability of DIFF in simultaneously managing both link usage and flow-table. Additionally, it decreases the packet latency and workload of the controller, thereby increasing network throughput relative to baseline schemes.

3.4.2. Control plane load balancing during controller failures

Software or hardware failure resulting in a master controller failure within a distributed multi-domain and multi-controller SDN LB initiates the immediate allocation of, predetermined slave controllers to the switches that were controlled by the failed master controller. Although this backup approach improves the reliability as well as the scalability of the SDN network, little or no research exists on how to plan the slave controller assignment following a master controller failure in the current work. In general, the present approach of assigning slave controllers is rigid and assign the closest controller to the switch as a slave as predetermined. A major flaw in this assignment scheme is exposed when there is a failure of the controller chain (Hu et al., 2018a, 2019).

In addition, a primary example is used in Fig. 8 to illustrate the phenomenon of controller chain failure. Fig. 8a, indicates three controllers (C1–C3) and six switches (S1–S6) on an SDN network. The network is divided into three domains (domains 1–3). A controller is capable of controlling a maximum of three switches. As shown in Fig. 8b, should C1 fail, the switches in domain 1 gets disconnected and assigned to C2, the nearest controller to C1. The position of C2 as a slave is also altered to master following the assignment of C1 switches in accordance with predetermined slave controller assignment protocol. However, due to a limit in control capacity (S3, S4), and (S1, S2) cannot be handled by C2 simultaneously. As illustrated in Fig. 8c, the flow requests sent by these four switches (S1 to S4) exhaust the capacity of C2, then C2 fails. Likewise, in Fig. 8d, four switches (S1 to S4) can also be assigned to C3 in the event of failure of C2. Also, C3 eventually fails when its capacity runs out. Finally, the network crashed following the failure of all the network controllers. This example illustrates how one controller's failure can result in the failure of another in a multi-controller SDN network, and the worst case, crashing of the entire network. This phenomenon is described as controller chain failure in the literature (Hu et al., 2019).

In (Hu et al., 2018a, 2019), dynamic slave controller assignment

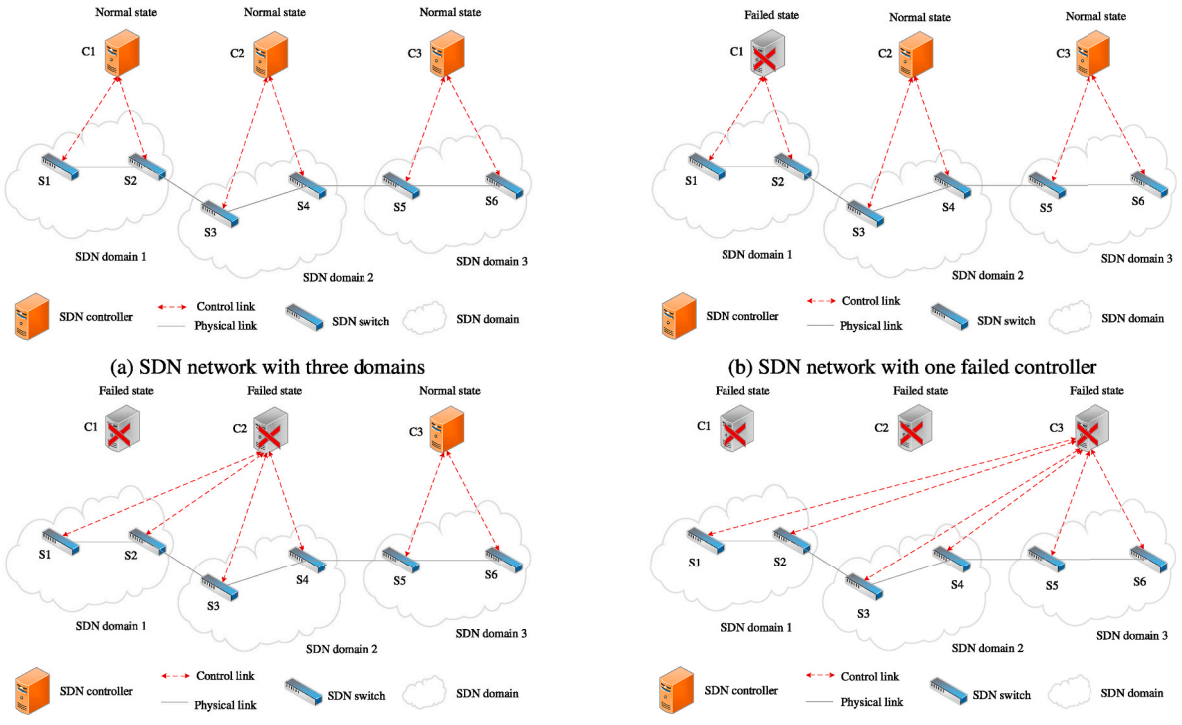

Fig. 8. The controller chain failure (Hu et al., 2019).

(DSCA) techniques were proposed for improved fault tolerance of the control plane and prevention of network crash through planned slave controller assignment to avoid controller chain failures. By defined the phenomenon of controller chain failure. To avoid the phenomenon, the assignment of a slave controller was formulated as a mixed multi-objective optimization problem that takes into account latency, LB, as well as robustness and proves its full complexity in the NP. Simulation results showed that DSCA could effectively ensure better fault-tolerance of the control plane's performance compared to other schemes. The occurrence and impact of controller failure on the SDN network also greatly minimized. Guo et al. (2019) introduced the joint control of resiliency maintenance and programmability of flows for software-defined WANs (SD-WANs) framework (RetroFlow) to achieve robust network control and flow programmability of the event of controller failures. RetroFlow maintains the normal operations of active controllers and the programmability of flows from offline switches by reducing the processing load of controllers from offline switches with commercial hybrid SDN switches that support switches operating in the legacy mode without controllers.

He et al. (He and Oki, 2020) introduced a preventive priority setting (PPS) model for handling LB against multiple controller failures in an SDN. This model has priority scales that is utilized for assigning a master controller to a switch. In the event of failure, the new master controllers are automatically obtained for switches whose current master controllers fail to recover control promptly according to the priority level. Priority sets are calculated at the beginning of network service to decrease the overall time consumption ratio in the worst-case scenario of failure. The empirical results showed that the new PPS model obtains the highest utilization relative to the existing work. Furthermore, the authors of (Guo et al., 2020a) introduced a programmability guardian technique to boost the programmability of offline flow paths while preserving low communication overhead. This technique configures the remapping flow controller to recover offline flows with identical path programmability, maximizes the overall programmability of the offline flows, and minimizes the overall overhead contact to monitor recovered flows.

3.4.3. *Switch migration mechanisms*

Traffic load should be transferred to the other controllers through the overload controller in charge of network flow to enhance the control plane capability of increasing response to switch requests. This concept is called a migration mechanism. This mechanism also enhances the availability of the controller (Cheng et al., 2015).

The requests for flow setup in diverse switches have to be allocated to suitable controllers dynamically. Although the OpenFlow protocol does not explicitly specify the switch migration protocol, the OpenFlow protocol clearly states that the controller has three roles: equal, slave, and master. There is a full control by the master/equal controller down to the switch, while the slave controller has only read permission to the switch (OpenFlow Protocol v1.5). When the switch connects to more than one controller, the switch is characterized with only a single master and numerous slave controllers. Whenever there is a need for the switch to migrate, it assigns the master role to a slave while it, the old master, becomes a slave.

Several previous works on switch migration (Dixit et al., 2013, 2014; Yu et al., 2016; Liang et al., 2014; Al-Tam and Correia, 2019; Sahoo and Sahoo, 2019; Filali et al., 2019) have proposed techniques to resolve LB's problem from different perspectives which includes: 1) selection of a migrated switch. 2) Selection of target controller. 3) Migration of switch. The problem of load oscillation may occur if the target controller and migrated switch are not correctly selected. One of the solutions to this problem between controllers is the switch selection technique proposed in (Yu et al., 2016). This technique cannot use the best of controller resources because it only considers the overloaded and target controller. Similarly, Liang et al. (2014) proposed an approach to address load oscillation problem but it could not enhance network LB because their focus was to improve the round-trip time and eliminate the switch and controller loads, respectively. To provide better improvement in the network in term of balancing, Dixit et al. (2014) proposed ElastiCon; a method which adopts a strategy called switch migration. ElastiCon performs a balance of the load of the controllers periodically in a pool. While this method provides a better performance, it creates a high network overhead problem by ignoring traffic dynamically. In

terms of time efficiency, the proposed techniques in (Dixit et al., 2013, 2014; Filali et al., 2019; Yu et al., 2016; Sahoo and Sahoo, 2019; Liang et al., 2014) select a switch to migrate most especially in LB; a process that does not allow the overloaded controller to become normal on time.

Zhou et al. (2014b) suggested an adaptive yet dynamic load balance algorithm (DALB)characterized by a distributed decision that operates as a module in the SDN controller. The module comprises of load measurement, load collection, and decision-maker. Every individual controller can ensure the measurement and collection of its load and other controllers to decide when there is a need for migration. Whenever the controller exceeds a certain threshold, the controller will make a load decision and ensure a balance on the load by selecting switches that are sufficient to guarantee this performance. In another work, Hu et al. (2017a) developed a strategy called an efficiency aware switch migration (EASM) for LB in multiple controllers. This method migrates switches according to the LB rate and cost of migration to enhance the efficiency and balance in controller load. Findings showed that this method enhances the controller response time, throughput, LB rate and migration cost.

Zhou et al. (2017) proposed an LB scheme that exhibits switch grouping. The researchers also developed a method that solves the issue of load oscillation between several controllers and can can make sure that the network balancing is improved using a switch selection algorithm and a formula for target selection. This technique ensures that the improvement gain remains stable even when the traffic conditions are dynamically higher. Similarly, to facilitate the improvement in time efficiency and design, it is possible to move a group of switches using a selection algorithm that is tightly controlled for switch grouping in a balancing process. The experimental results showed that their method addresses the load oscillation problem and ensure that network balancing can progress in a more time-efficient manner. Kyung et al. (2015) introduced a load distribution algorithm that works according to per-flows under multiple controllers. In particular, when a controller's capacity exceeds a threshold, incoming messages can be transferred by the controller to other controllers to avoid their blockage. Analytical findings show that the likelihood of being blocked in this scheme is low while ensuring higher power utilization of controllers than traditional schemes.

Xu et al. (2019) presented a balanced controller (BalCon) and BalConPlus, a two SDN switch migration approach for achieving LB between low-cost SDN controllers. BalCon is appropriate for scenarios where the network is not obliged to process switch requests serially. In (Sahoo et al., 2019b), an effective switch migration-based LB (ESMLB) method was implemented for the effective allocation of switches to underused controllers. In this method, a multicriteria decision-making process, i.e., the technique for order preference by similarity to an ideal solution (TOPSIS), was used among several alternatives for target controller selection. The sequence of the flow migration switch mechanism is illustrated in Fig. 9. As the switch tends to receive flow through the mobile node (MN) that does not probably match either of the existing entries, the switch decides and forward the packet-in a message via the controller1. Messages will be processed whenever the controller one is not overloaded, and this will eventually reply with a packet_Out/Flow_Mod (add) message to the switch. As the threshold is attained, controller 1 tend to migrate most of these messages to another controller (such as controller 2) which appeared not to be overloaded to handle the messages.

3.4.4. *Management SDN controller load*

The SDN controller architecture comprises one or more controllers useful for managing SDN load (i.e., scalability, reliability, availability) using distributed SDN controllers. The concept is founded on the premise that as multiple controllers are formed, they can share load within the network. Similarly, a controller can perform the work of

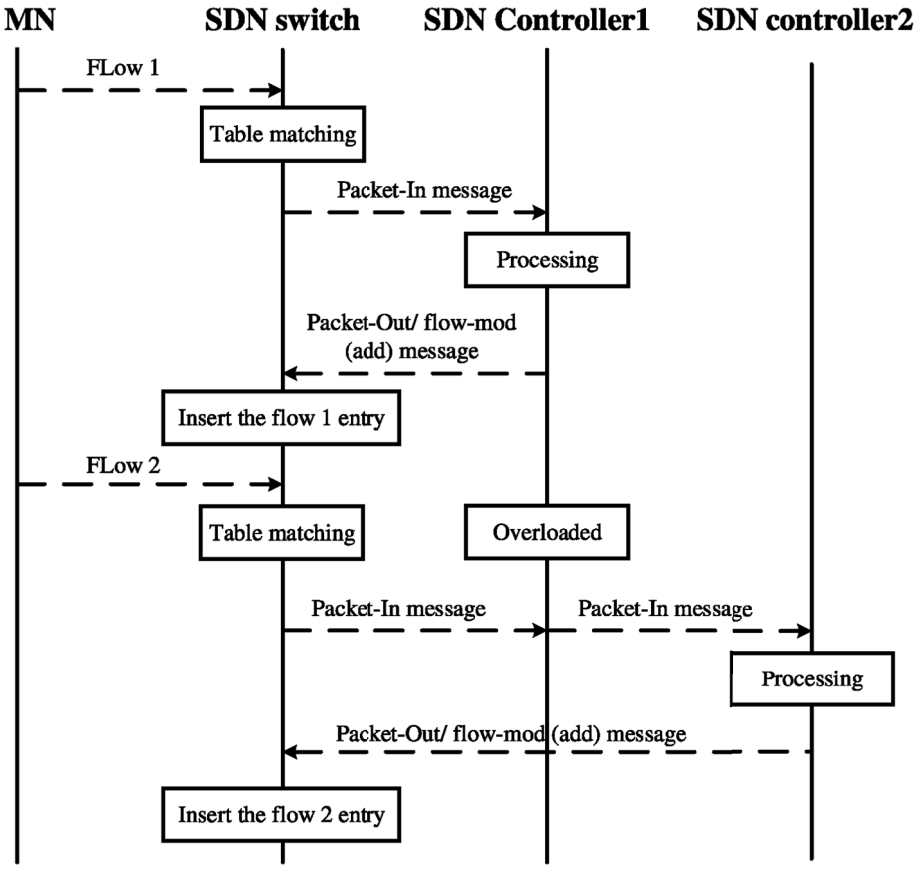

Fig. 9. Migration switch mechanism adoption from (Kyung et al., 2015).

another controller, especially when there is a crash. Most importantly, these procedures can be more complex such that the need to adopt a few methods from distributed SDN controllers systems may be required (Abdelaziz et al., 2017).

The scalability issue of LB in SDN can be viewed from two perspectives: i) the number of server pools controlled by a single controller, and ii) scalability of the control plane in SDN is usually estimated using the pair of the output (i.e., the number of flow requests handled per second) and the latency of the flow setup metrics (i.e., the delay in responding flow requests) (Zhou et al., 2014; Karakus and Durresi, 2017b; Bannour et al., 2018), a single controller that is physically controlled cannot meet the performance needs (in relation to those metrics) of big networks relative to small or medium-sized networks (see section 3.3). Some of the challenges related to scalability can be addressed by increasing the roles data plane to ease controllers' loading (see section 3.2). The key downside of this approach is that it applies some modifications to OpenFlow switch architecture. A more effective method is to ensure that the control plane modeled to reduce scalability challenges. Only one single SDN controller plays the role of handling incoming requests from SDN switches in a physically centralized control model. This may later hinder scalability and system general performance as the network gets bigger (Qiu et al., 2017). Thus, multiple controllers are used in a physically distributed control model for the creation of a network view that is logically coherent. This approach has been reported to manage controller bottleneck well as it greatly reduces control plane latencies while a manageable size of the control plane in the network (Abd-Allah et al., 2019).

The lack of reliability has been an extreme issue in SDN. However, decoupling the data-to-control plane would have a major effect on the reliability of the SDN control plane. The failure of the central controller would break the network entirely in a centralized SDN-based network. On the other hand, using multiple controllers in logically centralized but physically distributed controller architecture reduces a SPOF problem (Bannour et al., 2018). Similarly, handling network updates with different methods can result in the redundancy of controllers. In the active replication approach, client commands or requests are processed by multiple controllers in a coordinated and deterministic manner. This approach is often described as a simultaneous update of the replicated network state or state machine replication. The approach's critical challenge is to enforce a strict order of events to ensure efficient continuity between controller replicas. The replication method has the benefit of offering high durability at marginal downtime, which makes it an acceptable alternative in delay-intolerant situations. On the other hand, one controller (the primary) handles passive replication requests, referred to as primary/backup replication, updates the replicated status, and regularly alerts the other controller replicas (backups) to changes in status. While the passive replication scheme provides simplicity and lower overhead, it can create state inconsistencies (controller and switch) and generate additional delay if the primary controller fails (Spalla et al., 2016).

Most of the previous works on SDN (Abdelaziz et al., 2017; Berde et al., 2014; Dixit et al., 2013; Jimenez et al., 2014; Koponen et al., 2014; Tootoonchian and Ganjali, 2010; Muluye et al., 2019; Liu et al., 2020) have proposed distributed controller architecture to achieve reliability and scalability. Distributed controller architecture allows the network to scale-out without introducing bottlenecks or a SPOF. It also provides redundancy and fault tolerance for the SDN network. Distributed controller architecture can enhance scalability, fault tolerance, minimize latency, and availability in various SDN networks deployment scenarios (Abdelaziz et al., 2017). The architecture of the distributed SDN controller involves having multiple controllers (Eko OktianSang-Gon Lee, 2017). Distributed controllers can make available the deployments in SDN. Nevertheless, a peculiar problem with this approach remains the continuous inconsistency between distributed controllers. Due to inconsistencies between the controllers, network applications are handled incorrectly by the distributed controller because of the network state in the global view (Azodolmolky, 2013; Yang et al., 2014; Liu et al., 2020).

Additionally, the distributed controller generates issues associated with controller resource management, including the distribution of controller states, sharing of data, consistent and extended propagation delays between multiple controllers. Also, it limits the time of convergence and affects controller response to various networks. An elastically distributed SDN controller scheme was implemented in (Dixit et al., 2013). This scheme shrinks and enlarges the controller's pool based on network load. The proposed scheme was unable to provide clear information on the key challenge of resilient architecture. The researchers in (Zou et al., 2017) also proposed an active synchronization algorithm based on a multi-domain multi-controller periodic synchronization (PS) in SDN to handle the loading inconsistency among the controllers. The PS is based on temporal stiffness triggering (Levin et al., 2012). The proposed algorithm depends on events to get triggered rather than time. Therefore, controller synchronization is responsive to changes in the load of the controller. It also eliminates the overhead connection between time synchronization, thereby solving the inconsistency issue within the physically distributed device. Besides, the method avoids forwarding loops and enhances packet arrival rates. The experimental results showed that the active synchronization algorithm provides a very low packet loss rate, enhanced the performance of network load balance, and reduced synchronization overhead. When assessing the performance of SDN distributed controller architecture within data center networks, the major concerns remain the location and number of required controllers for given network topology. Liu et al. (2020) proposed adaptive adjustment and mapping controllers (AAMcon) to solve the following problems: very slow response time between switch and controller, difficulty in adapting the control plane to real network traffic load variations and controller selection.

3.4.5. Resource allocation and management

With the emergence of novel services and applications, the evolution of communication technologies has become a crucial task for the developing approaches to manage the resources of the networks with minimum human intervention. To establish management and control in an efficient and adaptive way, there is a need to look at both static and dynamic scenarios where a distributed controller LB approach is needed (Tuncer et al., 2015).

Centralized solutions achieved through the manager tend to have a global view in term of current situations necessary to compute new configurations based on dynamic traffic behavior, which is performed by the resource management approaches. Usually, centralized solutions have limitations in practice, especially when considering scalability and performance issues (Feng et al., 2014; Tao et al., 2015; Zehra and Shah, 2017). Therefore, SDN offers standard control interfaces and dynamic global views to solve these limitations. Forwarding of resource allocation and management in SDN is illustrated in Fig. 10; the data plane contains connected OpenFlow switches (OFS) while the control plane includes the OpenFlow controller (OFC). The centralized OFC computes the shortest path for each service based on the service's weight and global network view (such as the value or priorities of service). As the service weight become equal, the OFC then assigns bandwidth using the max-min procedure.

In term of allocating the resources of the network, SDN architectures can be grouped into three layers; infrastructure plane, which contains network elements, a control plane with a logically centralized controller, and an application plane which includes network control programs (Bailey et al., 2013). The deployment and operation of the control and application planes can be independently outside the network devices. The control applications use flow tables to manipulate network packet forwarding. The flow table hardware implementation is a limited capacity device, while the applications share the flow table space. Thus, the flow table is a unique network resource for showing control capability. The data plane switch performs matching of the packet header

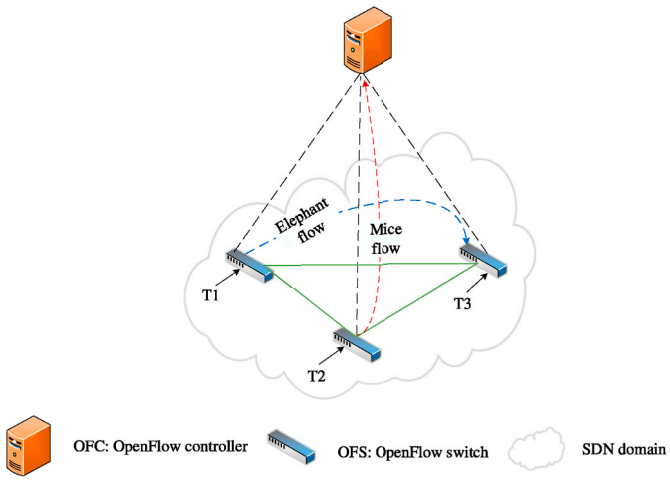

Fig. 10. Forwarding resource allocation and management in SDN.

with the flow table in the network resource scheduling mechanism. If there is a mismatch, the packet will be sent to the controller, where the control application will produce forwarding rules based on the packet and store them in the data plane flow table. Besides, service disruption can occur when trying to insert new flow entries into the full flow table (Tao et al., 2015).

Sherwood et al. (2010b) adopted the OpenFlow approach to allow the same hardware data plane to be disseminated across many logical networks. This method neglects the switch and link resources management. Several resources allocation management methods have been proposed in the existing literature (Tootoonchian and Ganjali, 2010; Koponen et al., 2010, 2014) to address some of the shortcomings of a single centralized controller model. A technique that uses information about the applications that run under the global data center to schedule and optimize the network bandwidth utilization with a centralized SDN-based traffic engineering system was introduced by (Jain et al., 2013). The applications are characterized according to the level of priority classes. Whenever an overloaded situation happens, the packets with low priority are rejected. The authors of (Jain et al., 2013) also introduced QoE-guided fair scheduling based on an SDN-based framework with a focus on networks that have limited resources. Similarly, Thi et al. (2017) developed a framework based on a QoS rate of allocation in OpenFlow switch networks for multi-class services. OpenFlow was also used to reduce the time of computation and control admission for the allocation rate. Given that the design is decentralized, the schemes can run on numerous parallel controllers; thereby ensuring the network's improved scalability. The controllers can manage more active and dynamics flow.

3.4.6. *Distributed controller placement*

Distributed controller architectures employ multiple controllers to solve the problems known with single SDN controller placement, e.g., availability (Hakiri et al., 2014; Lu et al., 2019; Li et al., 2019; Javadpour, 2020). Multiple controllers ensure decreased latency, improved scalability as well as tolerance to a fault. Nevertheless, the lookup overhead for communication among switches and numerous controllers is increased when using this architecture. A potential drawback of this approach is ensuring that the entire distributed network is maintained consistently. The network application tends to act inappropriately, particularly when the global network state's view is incoherent (Azodolmolky, 2013).

Guan et al. (2013) introduced a reliability-aware controller placement scheme that places a specified number of controllers in a particular physical network. The rate of valid control paths is used as a replacement metric to address the reliability issue. This technique incurred additional latency while handling the trade-off between latency and reliability. Latency propagation from the network switch and the controllers affect the controller's response time (Liao et al., 2017). Hu et al. (HU et al., 2012) proposed a centralized controller placement technique. The technique determines the number of SDN controllers needed and their placement in the network. In addition to the fact that this solution is heavily dependent on topology, it also becomes non-scalable as the network grows. Similar work by (Heller et al., 2012) addressed the placement of the controller to increase the control network's reliability. This approach optimized the trade-offs between latency and reliability. However, this method does not address the problem of dynamic load sharing among controllers when the network traffic changes. Alternatively, it focuses on the propagation delay. To improve the SDN network, the controller positioning needs to be considered to reduce the flow setup's average time under different traffic conditions (He et al., 2017).

Zhao et al. (Zhao and Wu, 2017) proposed a scalable placement architecture using linear integer programming and a heuristic algorithm to minimize the latency of connections. In addition, the technique established optimum controller positioning for wide area network topology with a focus on controller-to-controller as well as controller-to-switch delays. A min-cut, network partitioning and controller placement algorithm was introduced in (Beheshti and Zhang, 2012). The aim of this algorithm is to minimize the interruption between switch and controller links in the outgoing links without the use of backup. In the case where multiple controllers are required, this method cannot be applied. On the other hand, the authors in (Jalili et al., 2020) proposed a conceptual system for the placement of controllers while considering both the nature of the control plane and the interaction between the data and control planes. This framework is formulated as a multi-objective optimization problem that aims to optimize two objective functions for the least possible flow setup time and inter controllers' latency. Additionally, this solution proposed a time function model for flow setup that involves all of the placement metrics.

3.4.7. *Security issues in SDN load balancing*

Security and LB concept are critical issues that arise in the SDN enterprise market. The deployment and management of security solutions are difficult in a large multi-vendor environment (Benton et al., 2013; Scott-Hayward et al., 2013; Shu et al., 2016). From a security point of view, SDN monitors multiple levels of packets across the network. The deployment of security devices, e.g., firewalls, is very difficult in SDN networks because of SDN can use a center firewall to route all traffic. Using a single point for traffic flow is vulnerable, but it is easy to capture and analyze IDS and IPS in real-time (Khan et al., 2016b; Zhang et al., 2013; Dotcenko et al., 2014).

There are many security threats in an SDN network, such as trust establishment between the switch and the controller, securing and protecting the SDN controller, DoS/DDoS, long waiting queues, forensics and remediation, and the development of a robust policy framework (Chica et al., 2020; Yan et al., 2016; Khan et al., 2016a; Chourishi et al., 2015). Most of the previous works (Zhong et al., 2017b, 2019; Konglar and Somchit, 2018; Saldamli et al., 2019) have proposed a distributed controllers for SDN networks. However, all these studies neither considered security issues nor security encryption mechanisms. In (Selvi et al., 2016), a hierarchical network (COLBAS) controller LB technique was designed in which the controllers release the load periodically and coordinate each other to attain LB. This approach focused on how to incorporate LB for distributed controllers with no consideration for security in the distributed controller architecture (Porras et al., 2015).

A malicious controller is capable of infiltrating the network, get the network topology, unlawfully retrieve and alter network data, which may paralyze the network (Chua et al., 2015). Three architectures designed to be compatible with all existing frameworks such as OpenStack and OpenDaylight were proposed in (Chourishi et al., 2015). These architectures greatly improve the framework of the controller without interrupting the operations of the network. Likewise, some important issues that have to do with LB and safety are addressed in the

SDN framework. However, this framework is yet to be deployed on the SDN architecture. In another work, distributed controllers using a consistent and fault-tolerant data store that preserves the appropriate network and application state was proposed by (da Silva et al., 2020a). Unfortunately, this approach does not consider SDN safety requirements. It only aims to design, implement, and test a stable and consistent model for the DepSpace-based control plane while ensuring stable implementation of tuple space used to organize distributed processes. Experimental findings demonstrated the practical feasibility of the design proposed.

3.5. Performance metrics for load balancing techniques

In this subsection, we discuss the LB metrics for SDN networks. As earlier hinted, several LB techniques have been proposed in pertinent LB literature (e.g., (Zhong et al., 2017a; Chen-Xiao and Ya-Bin, 2016; Li et al., 2014; Li et al., 2019; Zeng et al., 2019; Abdelaziz et al., 2017; Chen et al., 2015b; Patil, 2018; Win et al., 2019; Li et al., 2020; Jamali et al., 2019; Abdelltif et al., 2018; Guo et al., 2014; Dixit et al., 2014; Wang et al., 2016a)). The metrics used in the implementation of these LB algorithms are summarized as follows:

3.5.1. Packet loss rate (P_r(Loss))

The packet loss rate is a common phenomenon that frequently occurs in data transmission devices. When the switches in the network get busy processing incoming packets, they are likely to drop some of the incoming packets. The switches become busy as a result of packet loss rate as well as the load condition of the path. The SDN controller will be able to ensure the accumulated number of transmitted packet Tx_{packet} are collected in OpenFlow switches ports, and also the cumulative number of the packet that is as received Rx_{packet} at respective OpenFlow switch ports. Therefore, the packet loss rate can then be calculated using Equation (1):

$$P_r(Loss) = \frac{Tx_{packet} - Rx_{packet}}{Tx_{packet}} \tag{1}$$

3.5.2. Throughput

This metric computes the sum of tasks completed in one unit time after performing LB. It determines the rate at which the computational work is done using an LB technique. The LB algorithm aims to achieve greater efficiency.

3.5.3. Average response time

One of the main metrics of LB techniques is the average response time. It is defined as the amount of time it takes for the user to retrieve the results of a request. Different variables that influence the response time include bandwidth, number of users accessing the network at the same time, number of requests and average processing time. A high number of requests per second must be processed to receive quicker responses.

3.5.4. Average end-to-end delay

The end-to-end delay of a flow is the time it takes for all packets to arrive at the destination node. It is calculated using the formula: (2).

$$T_{E2E} = T_{i\,End} - T_{i\,Start} \tag{2}$$

For the network, average end-to-end delay is the sum of all end-to-end delays divided by the total number of flows N_F.

$$Avg\,(T_{E2E}) = \frac{\sum_{i=1}^{N} T_{E2Ei}}{N_F} \tag{3}$$

3.5.5. Resource utilization ratio (RU ratio)

It is calculated as the resource's efficient use in processing the request, and the goal is to optimize the most critical resource in processing the request. The resources can be bandwidth utilization ratio, link, memory utilization, and processor. Furthermore, the resource utilization ratio represents a percentage of the overall connection capacity.

3.5.6. Overhead

Any combination of additional time, bandwidth, or any other required factors for the accomplishment of a given task is referred to as overhead. If packets are received by a node above its capacity Ex_{Packet}, overhead is generated at the node. Otherwise, if received packets are within range, no overhead is generated. Overhead at a node can be measured by using Equations (4) and (5):

$$Ex_{packet} = Rx_{packet} - Tx_{packet} \tag{4}$$

$$Ov_{Packet} = \frac{Ex_{packet}}{Packet\ threshold} \tag{5}$$

3.5.7. Transmission hop count (Hop Count)

The transmission hop count is considered as one of the significant factors for routing approach. The probability of congestion may be increased with a large number of transmission hop. In contrast, packet loss probability can be decreased as well as transmission delay in similar network condition if the packet transmitted along the path is made with fewer hops. As a result of the protection of the SDN controller's global network topology in the database, there is provision for several hops between a pair of switches; using source and destination switches as the query factor for searching the database.

3.5.8. Servers root mean squared error (RMSE)

It is a metric for measuring efficiency in LB techniques. A smaller RMSE has better efficiency. RMSE will be 0 if all servers are equally loaded.

3.5.9. Flow completion time

It is the primary metric used for determining flow transport efficiency in the case of large flow in data center networks. It can be defined as the duration it takes to complete a file transfer in a flow. The QoS specifications require a large array of network resources to complete the flow time. The FCT can include as amongst others the execution and migration times (i.e., time taken to move a switch is essential for the efficient operation of a control plane LB protocol). LB techniques aim to reduce the flow completion time to the minimum possible.

3.5.10. Types of traffic (EF, MF)

This involves differentiating the type of incoming traffic flow as to whether long-lived large flow (i.e., EF) or the short flow (i.e., MF). Based on previous measurements in the data center networks (Lin et al., 2014; Benson et al., 2010; Kandula et al., 2009; Greenberg et al., 2009), it has been found that 80% of the flows (i.e., MFs) are below 10 KB and lasted for a few milliseconds. In contrast, the top 10% of the large flows (i.e., EFs) account for the majority of the traffic. Any traffic that exceeds a specific threshold per unit time (e.g., 1 MBps) is classified as EFs (Lou et al., 2019). Given the high level of EFs in network traffic, their efficient control and rerouting can result in increased throughput of SDN network (Hamdan et al., 2020; Al-Fares et al., 2010). On the other hand, the competition for between MFs and EFs for limited network resources indicates that MFs do not have enough bandwidth (Wang et al., 2016b).

4. Discussion

A brief overview and comparative analysis of the various metrics used in the literature are explored from different perspectives of the various state-of-the-art LB techniques in this section. Firstly, it is necessary to address the problems associated with LB technology in order to improve their performance in SDN networks. Secondly, it is necessary to examine and categorize the identified problems and

solutions. In this paper, we studied the LB techniques for improving the SDN networks r to facilitate and ease research innovation for SDN researchers. With a focus on recent articles from both reputable journals and conferences, we have reviewed different issues to identify the most common performance enhancement approaches in different domains with a focus on LB, data plane LB techniques, control plane LB techniques, other aspects of data/control planes LB, performance measures for LB techniques, and lessons learned from LB techniques.

LB techniques in data plane are divide into server LB and link LB. The comparison of these techniques is presented in Tables 2 and 3, which shows a comparison of several SDN data plane LB solutions with different algorithms. The experimental environments focused on measurement metrics. Compared to traditional networks, SDN server LB techniques improve the efficiency of the LB server and can be useful in overcoming challenges associated with its implementation. LB offers many advantages, including website scalability, availability, manageability, and security. When a server or program fails, LB may also redirect the traffic to alternative servers. Server LB techniques, according to studies on static algorithms (Kaur et al., 2015; Zhang and Guo, 2014; Uppal and Brandon, 2010; Handigol et al., 2009b) allow only the controller to distribute the load to the servers without considering the server's efficiency. While dynamic LB algorithms (Srivastava and Pandey, 2020; Shang et al., 2013; Du and Zhuang, 2015; WANG et al., 2016; Chen et al., 2015a; Chou et al., 2014), which considers the actual load and the output at each node when managing task distribution, adjust the load of the nodes in a timely manner through the dynamic mechanism to ensure that the system runs smoothly over a long period of time. However, these approaches have several limitations that include the need to adapt burst traffics, as the need to adjust controllers load dynamically, and disregard for the types of services, amongst others.

Algorithms based on SBLB (Abdelltif et al., 2018) dynamically apply LB to different types of services in cloud-based SDN. These algorithms aim to optimize resource efficiency and reduce user response time. However, several limitations persist; such as in deploying adaptive LB based on traffic classification and in building application-aware LB. On the other hand, link LB techniques in data plane are based on heuristic algorithms such as (ECMP (Al-Fares et al., 2010), FSEM (Li et al., 2014), ACO (Dobrijevic et al., 2015; Wang et al., 2016a), GA (Jamali et al., 2019), GA-ACO (i.e., G-ACO) (Xue et al., 2019; Li et al., 2020)), and based on machine learning like (ANN (Chen-Xiao and Ya-Bin, 2016; Patil, 2018; Ruelas and Rothenberg, 2018; Rupani et al., 2020), DRL (Sun et al., 2019b, 2020b; Zhang et al., 2020; Yu et al., 2018)). They all perform comparatively better for large SDN networks (i.e., WAN), but with high processing and network overheads. Algorithms for dynamic LB in data plane traffic used for rerouting (Lin et al., 2016b; Hou et al., 2019; Zeng et al., 2019; Chahlaoui et al., 2019; da Silva et al., 2020b; Guo et al., 2016) work well for a comparatively limited SDN network, i.e., with a minimal number of target OpenFlow switches on the data plane. These algorithms lose their efficiency when the network exceeds a certain threshold or too large. Hence, the scalability of these methods cannot be ascertained.

Table 4 shows a comparison of several SDN control plane LB solutions with different controllers and algorithms. Although the focus was accorded to evaluation metrics in the experimental setup, the types of control plane LB were also considered. Most of these studies used different OpenFlow protocol versions. Logically-centralized and physically-distributed controller LB techniques, according to (Guo et al., 2014, 2020b; Sahoo et al., 2019a; Zhong et al., 2019; Hai and Kim, 2016; Tootoonchian and Ganjali, 2010), was proposed to overcome the idea of using replicated controllers to solve SPOF problems. Amongst the drawbacks of this approach is not triggering the disabled controllers before the main controller fails. Distributed controller LB techniques have garnered more research attention in recent years as they have emerged as a possible solution to the problems associated with centralized SDN architectures (performance bottlenecks, poor scalability etc.) (Zhou et al., 2014b; Zou et al., 2017; Zhong et al., 2019). Furthermore, one of the most distributed controllers LB technique is a hierarchical LB control techniques presented in (Lin et al., 2016a; Ma et al., 2017; Nayyer et al., 2019b) presumes vertical division of the network control plane into several layers) according to the needed services. Hierarchical controller-agents/sub-controllers LB mechanism aims to solve the synchronization problem, to reduce time consumption while boosting performance in the distributed controllers.

Moreover, the horizontal controller LB techniques also kind of distributed controller used in (Sun et al., 2020a; Hu et al., 2018b; Dixit et al., 2014; Yazici et al., 2014; Shang et al., 2018) introduced the horizontal division of the network into multiple sections, each controlled by a single controller whose responsibility is to manage the SDN sub-set switches. Among the numerous benefits of arranging controllers in a flat format are reduction control latency and improved resiliency. In a situation where the primary controller fails, a preselected backup controller begins accessing the present state of the system via shared consistent data that was stored as prevention against the cold-start (empty state) problem peculiar to the traditional passive replication approach. This ensures a smooth transfer of control to the new primary controller. On the other hand, the virtualization controller LB techniques according to previous studies (Chen et al., 2016; Jin et al., 2019), add a virtual layer over the infrastructure of the physical network. This layer provides virtualization of the network by building multiple virtual networks. These virtual networks consist of virtual resources like routers, switches, and other nodes that need to be managed and controlled in turn. The control is implemented via a transparent proxy that acts as a link between one side of the switch and several controllers at the other end.

Ensuring interoperability, reliability, scalability, and consistency, SDN control plane are among the top challenges that impede the design of a powerful, stable and high performance distributed SDN controller platforms. Reliability and scalability are major issues confronting fully centralized and distributed SDN control designs mainly because they are strongly affected by the number and positioning of multiple controllers as well as the configuration of the distributed control plane within the SDN network. Achieving the specifications for efficiency and availability typically come at the expense of maintaining a clear, unified network view needed for SDN application design and proper behavior. Consistency should also be considered among trade-offs involved in the design process of an SDN controller platform.

Table 5 illustrates how various works offer a comparative study of different SDN LB algorithms based on specific evaluation criteria. After analyzing the various LB techniques, it can be mentioned that different techniques have considered specific criteria for evaluation. Some of the researchers developed a single metric, while others found several metrics more appropriate. Fig. 11 graphically depicts the percentage of the SDN LB metrics considered in the reviewed articles. The most used metric is RU ratio, i.e., 52%, followed by a delay, i.e., 48%. Other metrics like throughput, packet loss, flow completion, response time, and overhead represent 36%, 31%, 21%, 16%, and 12%, respectively. Then, followed by RMSE and hop count that respectively, formed 7% of the articles. While the least used metric is a type of traffic is approximately 5%. Fig. 12 shows the year-wise pattern of SDN LB metrics according to the algorithms examined.

5. Lessons learned from the load balancing techniques for SDN

In this section, the main lessons learned from this survey are highlighted as follows:

- **Congestion awareness in SDN load balancing technique across the switches:** Data plane forwarding algorithms are easily deployable because they only use local congestion information from the OpenFlow switches in addition to not being equipped to modify the existing hardware switches and protocols. On the other hand, local information-based approaches perform smoothly when the topology

Table 5
Comparative analysis of load balancing techniques in SDN.

Reference	Category	Types of Algorithm	Packet Loss	Throughput	Response Time	Delay	RU Ratio	Overhead	Hop Count	RMSE	Flow Completion	Types of Traffic
Uppal and Brandon (2010)	Data plane LB	Load-based, random, round-robin				✓	✓					
Li et al. (2014)		FSEM, Top-k paths				✓						
Dobrijevic et al. (2015)		ACO									✓	
Chen-Xiao and Ya-Bin (2016)		BPANN	✓			✓	✓		✓			
Shang et al. (2013)		Random, round robin, SBLB			✓		✓					
Patil (2018)		ANN	✓			✓	✓	✓	✓			
Ruelas and Rothenberg (2018)		KDN based on MLP					✓			✓		
Abdelltif et al. (2018)		SBLB		✓	✓							
Jamali et al. (2019)		GPLB		✓		✓						
Srivastava and Pandey (2020)		LB based on channel capacity	✓		✓	✓		✓				
Chen et al. (2015a)		Round-robin, random, SBLB		✓			✓					
Du and Zhuang (2015)		Round-robin, random, SBLB		✓	✓		✓					
Zhong et al. (2017a)		Random, round-robin, LBBSR			✓		✓					
WANG et al. (2016)		Dynamic LB		✓	✓							
Wang et al. (2016a)		ACO	✓	✓		✓					✓	
Hou et al. (2019)		MPF	✓	✓								
Zeng et al. (2019)		LBLB	✓				✓					
Chahlaoui et al. (2019)		SDN LB	✓			✓	✓					
da Silva et al. (2020b)		Dynamic LB	✓	✓								
Yu et al. (2018)		DDPG with DROM		✓		✓						
Zhang et al. (2020)		CFR-RL				✓						
Guo et al. (2016)		AggreFlow					✓			✓		
Sun et al. (2020b)		SINET									✓	
Xue et al. (2019)		G-ACO	✓			✓					✓	
Wang et al. (2017)		ACO		✓		✓						✓
Rupani et al. (2020)		BPANN	✓			✓	✓		✓			
Li et al. (2020)		GA-ACO					✓					✓
Guo et al. (2014)	Control plane LB	LVS	✓				✓			✓		
(Hai and Kim, 2016)		Dynamic and adaptive LB		✓		✓		✓				
Ma et al. (2017)		LB mechanism based on MC		✓		✓	✓					
Chen et al. (2016)		Dynamic resource management				✓	✓	✓				
Shang et al. (2018)		Service-aware adaptive link LB		✓		✓	✓					
Guo et al. (2020b)		SeqAsn					✓				✓	
Nayyer et al. (2019b)		Laman				✓		✓			✓	
Jin et al. (2019)		TALON		✓								
Zhong et al. (2019)		SCPLBS					✓				✓	
Lin et al. (2016a)		QoS based on QAR	✓			✓	✓					
											✓	

(continued on next page)

Table 5 (*continued*)

Reference	Category	Types of Algorithm	Packet Loss	Throughput	Response Time	Delay	RU Ratio	Overhead	Hop Count	RMSE	Flow Completion	Types of Traffic
Yazici et al. (2014)		Distributed OpenFlow controller										
Dixit et al. (2014)		Elastic distributed controller architecture		✓	✓						✓	
Sahoo et al. (2019a)		Flow entry timeout	✓				✓					
Hu et al. (2018b)		BMS				✓						
Sun et al. (2020a)		MARVEL					✓					

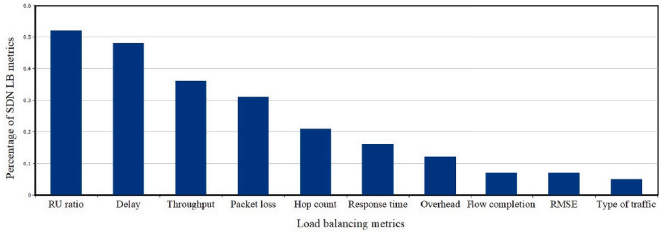

Fig. 11. Percentage of SDN load balancing metrics addressed in the surveyed techniques.

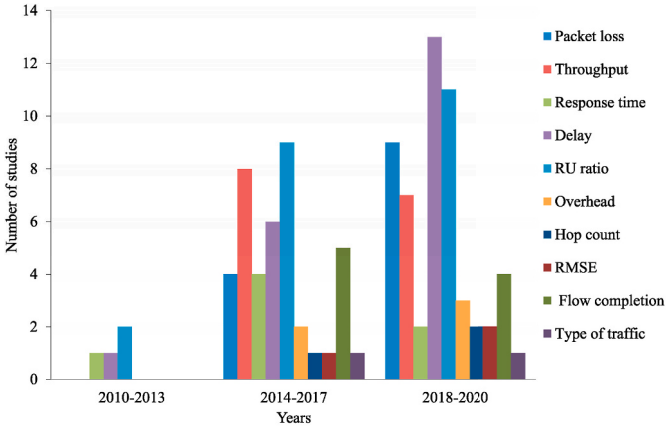

Fig. 12. The year-wise trend of addressed SDN load balancing metrics.

of the network is symmetric, and the length of the flow deviation is small. However, network topologies are typically asymmetric, and the flow distribution is often heavy-tailed. Thus, local information-based schemes may even have worse performance than static plans like ECMP (Al-Fares et al., 2010). Fortunately, modern approaches, like ACO (Dobrijevic et al., 2015; Wang et al., 2016a), have shown that local algorithms with OpenFlow switches do have the potential of achieving optimal efficiency and outperforming global congestion-aware approaches.

- **Congestion awareness in SDN load balancing technique across the controllers:** The congestion-aware controller system uses a centralized controller to collect real-time link usage and traffic information or adopts a distributed network for transmitting congestion information between the OpenFlow switch and the host. In LB methods, global congestion information is considered necessary because traffic, as well as topology, are typically asymmetric, hence, susceptible to degradation in performance due to the use of only local information. However, the cost of gathering global congestion information in an SDN network is high. Centralized methods use controllers to collect data, but they typically operate in full control loops during which congestion might have subsided. Distributed technologies use tracking or piggy-backing to relay information globally. This is capable of relaying congestion information in a timely fashion but at the cost of increased overhead.
- **Centralized load balancing in SDN networks:** Decision on whether to choose a centralized or distributed architecture is the first thing to consider. Generally, unified architectures are designed for modern programmable networks, such as SDN. A globally centralized controller is deployed in these architectures to collect traffic matrices and connection information on usage in real-time as well as to change traffic or reschedule track assignments. The benefits of using a controller include a fast understanding of traffic dynamics and a reliable analysis of congestion that has arisen within the network. Furthermore, the globally centralized controller can help to manage the scheme identification load of asymmetric topologies triggered by communication failures and re-planning decisions on time. The centralized design's drawback is that the controller becomes an extensive bottleneck network which limits performance.
- **Distributed load balancing in SDN networks:** Considering the limitations of centralized controllers, distributed solutions offer improved performance by leveraging distributed switches or hosts for decision making. Distributed Algorithms often apply their mechanisms to switches or end hosts using distributed protocols to collect and process global congestion and link failure information. Decentralized solutions can handle traffic faster and easier to deploy on a large network. However, when it comes to detailed information about traffic patterns and global link utilization, distributed algorithms may not optimally balance the load due to difficulty in handling link failures in a timely manner. Hence, distributed algorithms use congestion or ACK to transmit congestion information between switches or hosts; thus incurring additional overhead as well as limitation in the scalability of the algorithm.
- **Network-based load balancing in SDN:** In-network strategies are implemented at end-hosts instead of the transport layer of the network structure. As the transmission protocol on the end host is already complex enough, the LB scheme implemented in the network structure avoids further complexity to the end host. However, In-network methods have to learn flow numbers and count packets if they need traffic information of applications; thus incurring additional costs.
- **Host-based load balancing in SDN:** A host-based approach has the advantage of getting traffic information directly from running applications on the server. Whatever the network architecture, host-based servers can be distributed across several networks. This is feasible, even in broader topologies. The weakness of the host-based approach is that they can make the end-host transport stack too complicated, thus raising the end-host overhead. Besides, some SDN applications bypass the kernel network making this infeasible for using host-based approaches.

6. Challenges and future research issues

Several problems exist in the research process associated with the LB mechanisms. These problems can be solved by extra development and optimization of SDN research. We proffer several open future research directions on SDN LB mechanisms in this section:

1. **Load balancing technique to handle high controller load in data plane:** Some techniques have been proposed in the literature to handle high controller load. Genetic-based LB algorithm (Chou et al., 2014) optimal scheme was also put forward as a solution to avoid the high bottleneck in a single controller while saving network cost. Similarly, another approach called FSEM (Li et al., 2014) used Top-K algorithm to choose the shortest path. As a result, accurate real-time based on the load condition of each path cannot be demonstrated based on this technique as it only considers hop count, which may exclude some optimal paths. ACO was proposed in (Dobrijevic et al., 2015) as an estimation model to utilize QoE and ensure the user QoE is maximized for the services associated with multimedia. This technique only considers the weighted graph, which is not sufficient to obtain the shortest path. To predict how integrated load that has to do with different paths decides to choose a path with the least load path, BPANN was introduced by (Chen-Xiao and Ya-Bin, 2016). This path represents the data-flow transmission path. Certain characteristics exhibited by the BPANN are associated with these objectives that comprise of utilization ratio of the bandwidth, transmission latency, packet loss rate and transmission hops. The main drawback of this algorithm is that it neglects the various types of services which impede finding of the exact shortest path. Therefore, there is the need to address the established limitations of applying LB dynamically on application.
2. **Dynamic load balancing technique for multiple controllers:** The deployment of multiples controllers demands the consideration of traffic engineers; especially in a large-scale SDN network as this is an important subject in network performance optimization. Thus, a dynamic LB in the SDN is needed to adapt burst traffics as well as to adjust controller load without neglecting effective traffic balancing dynamically.
3. **Hierarchical controller load balancing:** Centralized SDN control using a single has been proven to be insufficient in exchanging large amounts of data and result in SPOF when the controller fails. Therefore, the entire network may collapse because all forwarding decisions depend on a single controller (Yazici et al., 2014; Akyildiz et al., 2016; Nayyer et al., 2019b). Unlike the centralized decision controller, the super controller is responsible for maintaining the global controller load information table but with an extended LB time (Dixit et al., 2013). LB decisions implemented in DALB are highly time-consuming when performing distributed LB (Zhou et al., 2014b). The previous synchronization of multiple controllers is formed based on PS (Levin et al., 2012), which ignores potential inconsistencies in global networks and further generates large amounts of packet loss and poor load balance. Several research on the distributed controller has focused on the synchronization overhead of the controllers in different domains without implementing an LB solution (Guo et al., 2014). The advantage of the synchronization overhead in the PS lies in the reduced synchronization time; although network performance degrades over time.
4. **Network virtualization within multiple controllers:** Network virtualization of essential infrastructures is an abstraction that can adjust the workloads as well as the flexibility of the resources to meet various requirements of services (Yi et al., 2018; Chen et al., 2016; Sun et al., 2019a). The integration of the SDN with network functions virtualization (NFV) could eventually trigger most innovative designs that are capable of exploiting the full benefits of both paradigms (Duan et al., 2016; Abdelaziz et al., 2016). The recent increase in research is now geared towards the concept of virtualization of infrastructure. This method which requires that the virtualization process be implemented on numerous SDN controllers, has become a significant research topic that requires detailed investigation.
5. **Controller placement techniques:** The placement of controllers directly impacts network performance (Heller et al., 2012). Sub-optimal controller placements affect flow rule setup latency and thus, result in control plane overhead, delay in controller-switch communication, the resilience of the fault tolerance as well as controller-controller communication delay. Although some studies have attempted to address some of these problems in the literature (Cello et al., 2017; Heller et al., 2012; Dvir et al., 2018; Obadia et al., 2015; Jimenez et al., 2014; Xiao et al., 2014; Hu et al., 2013; Lu et al., 2019; Jalili et al., 2019; Javadpour, 2020), this is an on-going research area with several issues that demand further readdressing by researchers.
6. **Flow rule setup latency:** In trying to ensure device plan scalability, this can result in other problems such as delay in the new flow rule setup (He et al., 2015). As earlier stated, there are two modes of setting up a new flow rule, namely; proactive and reactive modes. In the flow rules, the proactive mode is usually insufficient in enforcing any latency from the controller's perspective of the flow rules. The response time of the controller becomes crucial in a reactive mode. However, it may become impossible for controllers with flow rules setups to meet the necessities of few applications like the fast failover as well as the latency-sensitive flows of reactive routing. Consequently, such control planes may lack the requisite scalability for ensuring that the needs of the network are satisfied. Nevertheless, by imposing a large number of switches and controllers such as CPU and memory, the delay can be reduced as well by decentralizing a few control functions within the switches. To reduce latency in the data plane LB, it is possible to use link migration approaches for even distribution of load among OpenFlow switches. This will maintain the required QoS in SDN networks with latency in data plane flow rule setup.
7. **Network management challenges for software-defined based internet of things (SD-IoT):** It is predicted that, in a decade or so, through the power of IoT technology, billions of devices will be in use worldwide (Bera et al., 2016; Srinidhi et al., 2019). Thus, massive data is expected to be generated by devices that need timely and efficient processing. Accordingly, network management is an essential element in handling such a massive range of devices and the vast knowledge they produce. Therefore, suitable technologies are necessary to distribute and monitor network traffic flows for LB and network delay minimization. The SDN-based technology will meet these requirements since it has a centralized global view of the network. Thus, IoT network control like LB, fine-grained traffic routing, and increased bandwidth efficiency can be extended to the SDN-based technologies (Bera et al., 2017; Montazerolghaem and Yaghmaee, 2020).
8. **Data plane fault tolerance and low-latency load balancing for SD-WAN:** Programmability in SDN provides opportunities to design custom, quick and effective adaptive routing schemes for low end-to-end latency, as well as failure recovery mechanisms that can be accomplished with low overhead communication and low interaction of controllers (Zhao et al., 2020a; Qureshi et al., 2020; Michel and Keller, 2017). However, in the presence of connection failure or congestion, which are scenarios typical in SD-WANs, it remains unclear how to pick better paths in real-time.
9. **Energy-efficient networking based load balancing technique:** Turning the network components on and off based on traffic characteristics and using connections in traffic-aware

solutions decreases energy consumption. But deciding the collection of network components to dynamically turn on or off without affecting QoS and network output is an NP-hard problem (Zhao et al., 2020b; Kurroliya et al., 2020; Assefa and Özkasap, 2019; Guo et al., 2016). The trade-off between energy savings and network efficiency should be seen as a practical solution in this field. Through SDN-based LB, a method to reduce energy consumed by network flow scheduling is introduced at the data center network. However, lower flow-sizes get higher priority compared with higher flow-sizes. Higher flow-sizes will also not be supported during their deadline if there are much lower flow-size. Minimizing energy consumption in networks (e.g., SDN based data centers, SD-IoT, SD-WAN) is a critical factor in supporting green technology. Therefore appropriate strategies for energy-efficient LB-based networks need to be proposed.

10. **Security challenges:** Availability is a security problem which is closely linked to protection and scalability. Most of the SDNs security threats are designed for compromising the control plane availability. Therefore, it is recommended to use multiple controllers. However, simply adding multiple controllers can cause cascade failures in the controller, as demonstrated in (Yao et al., 2013). Therefore, security and scalability in SDN must be correlated with the implementation of a secure architecture that ensures the high availability of the control plane (Ahmad et al., 2015; Khan et al., 2020). Several security experts assert that SDN is native to vulnerability because it eliminates hardware obstacles such as firewalls that are secured (Kreutz et al., 2013; Chica et al., 2020). Protecting the controller and building trust between controllers is a key issue. Likewise, DDoS and long-awaited queues are SDN challenges (Shaghaghi et al., 2020; Polat et al., 2020). Providing LB with improved protection is essential to transitioning to SDN. Additionally, it is understood that an OpenFlow controller's functionality and operating set are most likely limited. Therefore, the lack of scalability in SDN will allow targeted attacks to cause control plane saturation by flooding communication between the controller and the switch (Shin et al., 2013; Ahmad et al., 2015). Moreover, security and scalability in SDN need to be linked to build stable SDN architectures that ensure a highly accessible control plane. LB and link synchronization strategies may be used for the data plane to share loads between OpenFlow switches in equal measure. This will not only help sustain the necessary QoS but also mitigate flooding severity and saturation attacks on data planes.

7. Conclusion

In SDN networks, many load balancing techniques can be used to improve network performance. This is because of the global view of the resources increases as a result of the SDN controller, and the implementation of the application requirements knowledge to distribute the traffic load.

In this article, we present a comprehensive survey of load balancing techniques in SDN. Firstly, we discussed the SDN architecture associated with the load balance. This is followed by a brief discussion on LB objectives, such as to minimize response time, optimize resource use, avoiding bottlenecks, and maximize the throughput. The LB techniques were further classified by dividing the LB into data plane LB and control plane LB. Then, the data plane LB is further divided into the server LB, and the link LB. Moreover, the control plane LB that includes logically-centralized/physically-distributed LB distributed LB, which is divided into (hierarchical, horizontal), and virtualization controller LB is discussed at length. Additionally, we provide a brief discussion on the other existing aspects of data/control planes LB such as switch flow table LB, LB during controller failures, the migration switch mechanism, managing SDN controller load, resource allocation, and management in SDN, distributed controller placement as well as security issues. Some metrics that are related to the performance of LB in SDN like packet loss rate, throughput, average end-to-end delay, the average response time, root mean squared error, resource utilization ratio, overhead, transmission hop count, flow completion time, and types of traffic have also been presented in fine detail. Furthermore, a comparative analysis of the surveyed techniques and the lessons learned from LB in SDN are summarized. Finally, we expose the open issues and future directions for further research that could improve the wide acceptance of the emerging application of load balance in SDN.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abd-Allah, A.G.A., Adly, A.S., Ghalwash, A.Z., 2019. Scalability between flow tables & multiple controllers in software defined networking. International Journal of Computer Science and Information Security (IJCSIS) 17 (2019). https://doi.org/10.1109/WAINA.2014.153.

Abdelaziz, A., Ang, T.F., Sookhak, M., Khan, S., Vasilakos, A., Liew, C.S., Akhunzada, A., 2016. Survey on network virtualization using OpenFlow: taxonomy, opportunities, and open issues. Ksii Transactions on Internet & Information Systems, p. 10, 2016.

Abdelaziz, A., Fong, A.T., Gani, A., Garba, U., Khan, S., Akhunzada, A., Talebian, H., Choo, K.K.R., 2017. Distributed controller clustering in software defined networks. PloS One 12, e0174715.

Abdelltif, A.A., Ahmed, E., Fong, A.T., Gani, A., Imran, M., 2018. SDN-based load balancing service for cloud servers. IEEE Commun. Mag. 56, 106–111.

Adami, D., Giordano, S., Pagano, M., Santinelli, N., 2014. Class-based traffic recovery with load balancing in software-defined networks. In: 2014 IEEE Globecom Workshops (GC Wkshps). IEEE, pp. 161–165.

Agarwal, S., Kodialam, M., Lakshman, T., 2013. Traffic engineering in software defined networks. In: 2013 Proceedings IEEE INFOCOM. IEEE, pp. 2211–2219.

Ahmad, I., Namal, S., Ylianttila, M., Gurtov, A., 2015. Security in software defined networks: a survey. IEEE Commun. Surv. Tutor. 17, 2317–2346.

Akbar Neghabi, A., Jafari Navimipour, N., Hosseinzadeh, M., Rezaee, A., 2019. Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. Int. J. Commun. Syst. 32, e3875.

Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W., 2014. A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Network. 71, 1–30.

Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W., 2016. Research challenges for traffic engineering in software defined networks. IEEE Network 30, 52–58.

Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A., 2010. Hedera: dynamic flow scheduling for data center networks. In: NSDI, 1919.

Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Parulkar, G., Salvadori, E., Snow, B., 2014. Openvirtex: make your virtual SDNs programmable. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 25–30.

Al-Tam, F., Correia, N., 2019. Fractional switch migration in multi-controller software-defined networking. Comput. Network. 157, 1–10.

Alakeel, A.M., 2010. A guide to dynamic load balancing in distributed computer systems. Int. J. Comput. Sci. Inf. Secur. 10, 153–160.

Alsaeedi, M., Mohamad, M.M., Al-Roubaiey, A.A., 2019. Toward adaptive and scalable OpenFlow-SDN flow control: a survey. IEEE Access 7, 107346–107379.

Assefa, B.G., zkasap, ., 2019. A survey of energy efficiency in SDN: software-based methods and optimization models. J. Netw. Comput. Appl. 137, 127–143.

Azodolmolky, S., 2013. Software Defined Networking with OpenFlow. Packt Publishing Ltd.

Badirzadeh, A., Jamali, S., 2018. A survey on load balancing methods in software defined network. Network. Commun. Eng. 10, 21–27.

Bailey, S., Bansal, D., Dunbar, L., Hood, D., Kis, Z.L., MackCrane, B., Maguire, J., Malek, D., Meyer, D., Paul, M., et al., 2013. SDN Architecture Overview. Open Networking Foundation, 2013, Ver 1.

Bannour, F., Souihi, S., Mellouk, A., 2018. Distributed SDN control: survey, taxonomy, and challenges. IEEE Commun. Surv. Tutor. 20, 333–354.

Bari, M.F., Roy, A.R., Chowdhury, S.R., Zhang, Q., Zhani, M.F., Ahmed, R., Boutaba, R., 2013. Dynamic controller provisioning in software defined networks. In: Network and Service Management (CNSM), 2013 9th International Conference on. IEEE, pp. 18–25.

Beheshti, N., Zhang, Y., 2012. Fast failover for control traffic in software-defined networks. In: Global Communications Conference (GLOBECOM), 2012 IEEE. IEEE, pp. 2665–2670.

Belgaum, M.R., Musa, S., Alam, M.M., Suud, M.M., 2020. A Systematic Review of Load Balancing Techniques in Software-Defined Networking. IEEE Access, 2020.

Benamrane, F., Benaini, R., et al., 2017. An east-west interface for distributed SDN control plane: implementation and evaluation. Comput. Electr. Eng. 57, 162–175.

Benson, T., Anand, A., Akella, A., Zhang, M., 2010. Understanding data center traffic characteristics. Comput. Commun. Rev. 40, 92–99.

Benton, K., Camp, L.J., Small, C., 2013. OpenFlow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 151–152.
Bera, S., Misra, S., Obaidat, M.S., 2016. Mobility-aware flow-table implementation in software-defined iot. In: 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, pp. 1–6.
Bera, S., Misra, S., Vasilakos, A.V., 2017. Software-defined networking for internet of things: a survey. IEEE Internet of Things J. 4, 1994–2008.
Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., et al., 2014. Onos: towards an open, distributed SDN os. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 1–6.
Blial, O., Ben Mamoun, M., Benaini, R., 2016. An overview on SDN architectures with multiple controllers. J. Comput. Netw. Commun. 2016.
Boero, L., Cello, M., Garibotto, C., Marchese, M., Mongelli, M., 2016. BeaQoS: load balancing and deadline management of queues in an OpenFlow SDN switch. Comput. Network. 106, 161–170.
Bozakov, Z., Sander, V., 2013. OpenFlow: a perspective for building versatile networks. In: Network-Embedded Management and Applications. Springer, pp. 217–245.
Canini, M., Kuznetsov, P., Levin, D., Schmid, S., 2013. Software transactional networking: concurrent and consistent policy composition. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 1–6.
Cello, M., Xu, Y., Walid, A., Wilfong, G., Chao, H.J., Marchese, M., 2017. Balcon: a distributed elastic SDN control via efficient switch migration. In: Cloud Engineering (IC2E), 2017 IEEE International Conference on. IEEE, pp. 40–50.
Chahlaoui, F., Dahmouni, H., 2020. A taxonomy of load balancing mechanisms in centralized and distributed SDN architectures. SN Comput. Sci. 1, 1–16.
Chahlaoui, F., El-Fenni, M.R., Dahmouni, H., 2019. Performance analysis of load balancing mechanisms in SDN networks. In: Proceedings of the 2nd International Conference on Networking. Information Systems & Security, pp. 1–8.
Chen, W., Shang, Z., Tian, X., Li, H., 2015a. Dynamic server cluster load balancing in virtualization environment with OpenFlow. Int. J. Distributed Sens. Netw. 11, 531538.
Chen, W., Shang, Z., Tian, X., Li, H., 2015b. Dynamic server cluster load balancing in virtualization environment with OpenFlow. Int. J. Distributed Sens. Netw. 11, 531538.
Chen, J.L., Ma, Y.W., Kuo, H.Y., Yang, C.S., Hung, W.C., 2016. Software-defined network virtualization platform for enterprise network resource management. IEEE Trans. Emerg. Top. Comput. 4, 179–186.
Chen-Xiao, C., Ya-Bin, X., 2016. Research on Load Balance Method in SDN, 2016.
Cheng, G., Chen, H., Wang, Z., Chen, S., 2015. Dha: distributed decisions on the switch migration toward a scalable SDN control plane. In: IFIP Networking Conference (IFIP Networking). IEEE, pp. 1–9, 2015.
Chica, J.C.C., Imbachi, J.C., Botero, J.F., 2020. Security in SDN: a comprehensive survey. J. Netw. Comput. Appl. 102595.
Chou, L.D., Yang, Y.T., Hong, Y.M., Hu, J.K., Jean, B., 2014. A genetic-based load balancing algorithm in OpenFlow network. In: Advanced Technologies, Embedded and Multimedia for Human-Centric Computing. Springer, pp. 411–417.
Chourishi, D., Miri, A., Mili, M., Ismaeel, S., 2015. Role-based multiple controllers for load balancing and security in SDN. In: Humanitarian Technology Conference (IHTC2015). 2015 IEEE Canada International, IEEE, pp. 1–4.
Chua, R.L., Pearce, A.K., Palmer, M., 2015. Authentication for software defined networks. US Patent 9,038,151.
Clayman, S., Mamatas, L., Galis, A., 2016. Efficient management solutions for software-defined infrastructures. In: Network Operations and Management Symposium (NOMS). 2016 IEEE/IFIP, IEEE, pp. 1291–1296.
CN, S., et al., 2019. A proactive flow admission and re-routing scheme for load balancing and mitigation of congestion propagation in SDN data plane. Int. J. Comput. Network. Commun. 10 (2019).
Curtis, A.R., Kim, W., Yalagandula, P., 2011a. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection. In: INFOCOM, 2011 Proceedings IEEE. IEEE, pp. 1629–1637.
Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S., 2011b. Devoflow: scaling flow management for high-performance networks. In: Proceedings of the ACM SIGCOMM 2011 Conference, pp. 254–265.
da Silva, J.P., Alchieri, E., Bordim, J., Costa, L., 2020a. A secure and distributed control plane for software defined networks. In: International Conference on Advanced Information Networking and Applications. Springer, pp. 994–1006.
da Silva, L.S., Storck, C.R., de LP Duarte-Figueiredo, F., 2020b. A Dynamic Load Balancing Algorithm for Data Plane Traffic.
Daraghmi, E.Y., Yuan, S.M., 2015. A small world based overlay network for improving dynamic load-balancing. J. Syst. Software 107, 187–203.
Devapriya, R.D., Gandhi, S.I., 2020. Enhanced load balancing and QoS provisioning algorithm for a software defined network. In: 2020 International Conference on Emerging Trends in Information Technology and Engineering (Ic-ETITE). IEEE, pp. 1–5.
Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R., 2013. Towards an elastic distributed SDN controller. In: ACM SIGCOMM Computer Communication Review. ACM, pp. 7–12.
Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R.R., 2014. Elasticon; an elastic distributed SDN controller. In: Architectures for Networking and Communications Systems (ANCS), 2014 ACM/IEEE Symposium on. IEEE, pp. 17–27.
Dobrijevic, O., Santl, M., Matijasevic, M., 2015. Ant colony optimization for qoe-centric flow routing in software-defined networks. In: Network and Service Management (CNSM), 2015 11th International Conference on. IEEE, pp. 274–278.
Dotcenko, S., Vladyko, A., Letenko, I., 2014. A fuzzy logic-based information security management for software-defined networks. In: Advanced Communication Technology (ICACT), 2014 16th International Conference on. IEEE, pp. 167–171.
Du, Q., Zhuang, H., 2015. OpenFlow-based dynamic server cluster load balancing with measurement support. J. Commun. 10, 16–21.
Duan, Q., Ansari, N., Toy, M., 2016. Software-defined network virtualization: an architectural framework for integrating SDN and nfv for service provisioning in future networks. IEEE Network 30, 10–16.
Dvir, A., Haddad, Y., Zilberman, A., 2018. Wireless controller placement problem. In: Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual. IEEE, pp. 1–4.
Eko Oktian, Yustus, SangGon Lee, H.L.J.L., 2017. Distributed SDN controller system: a survey on design choice. Comput. Network. 121, 100–111.
Feamster, N., Rexford, J., Zegura, E., 2013. The road to SDN. Queue 11, 20.
Feng, T., Bi, J., Wang, K., 2014. Joint allocation and scheduling of network resource for multiple control applications in SDN. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, pp. 1–7.
Filali, A., Cherkaoui, S., Kobbane, A., 2019. Prediction-based switch migration scheduling for SDN load balancing. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.
Fratczak, T., Broadbent, M., Georgopoulos, P., Race, N., 2013. Homevisor: adapting home network environments. In: Software Defined Networks (EWSDN), 2013 Second European Workshop on. IEEE, pp. 32–37.
Fundation, O.N., 2012. Software-defined networking: the new norm for networks. Open Netw. Fundation White Paper 2, 2–6.
Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S., 2009. Vl2: a scalable and flexible data center network. In: ACM SIGCOMM Computer Communication Review. ACM, pp. 51–62.
Guan, X., Choi, B.Y., Song, S., 2013. Reliability and scalability issues in software defined network frameworks. In: Research and Educational Experiment Workshop (GREE), 2013 Second GENI. IEEE, pp. 102–103.
Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S., 2008. Nox: towards an operating system for networks. Comput. Commun. Rev. 38, 105–110.
Guo, Z., Su, M., Xu, Y., Duan, Z., Wang, L., Hui, S., Chao, H.J., 2014. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. Comput. Network. 68, 95–109.
Guo, Z., Xu, Y., Cello, M., Zhang, J., Wang, Z., Liu, M., Chao, H.J., 2015. Jumpflow: reducing flow table usage in software-defined networks. Comput. Network. 92, 300–315.
Guo, Z., Hui, S., Xu, Y., Chao, H.J., 2016. Dynamic flow scheduling for power-efficient data center networks. In: 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS). IEEE, pp. 1–10.
Guo, Z., Liu, R., Xu, Y., Gushchin, A., Walid, A., Chao, H.J., 2017. Star: preventing flow-table overflow in software-defined networks. Comput. Network. 125, 15–25.
Guo, Z., Xu, Y., Liu, R., Gushchin, A., Chen, K.y., Walid, A., Chao, H.J., 2018. Balancing flow table occupancy and link utilization in software-defined networks. Future Generat. Comput. Syst. 89, 213–223.
Guo, Z., Feng, W., Liu, S., Jiang, W., Xu, Y., Zhang, Z.L., 2019. Retroflow: maintaining control resiliency and flow programmability for software-defined wans. In: 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS). IEEE, pp. 1–10.
Guo, Z., Dou, S., Wenchao, J., 2020a. Improving the path programmability for software-defined wans under multiple controller failures. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). IEEE, pp. 1–8.
Guo, Z., Zhang, S., Feng, W., Wu, W., Lan, J., 2020b. Exploring the role of paths for dynamic switch assignment in software-defined networks. Future Generat. Comput. Syst. 2020.
Hai, N.T., Kim, D.S., 2016. Efficient load balancing for multi-controller in SDN-based mission-critical networks. In: Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on. IEEE, pp. 420–425.
Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T., 2014. Software-defined networking: challenges and research opportunities for future internet. Comput. Network. 75, 453–471.
Hamdan, M., Mohammed, B., Humayun, U., Abdelaziz, A., Khan, S., Ali, M.A., Imran, M., Marsono, M., 2020. Flow-aware elephant flow detection for software-defined networks. IEEE Access 8, 72585–72597.
Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., Johari, R., 2009a. Plug-n-serve: load-balancing web traffic using OpenFlow. ACM Sigcomm Demo 4, 6.
Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., Johari, R., 2009b. Plug-n-serve: load-balancing web traffic using OpenFlow. ACM Sigcomm Demo 4, 6.
Hassas Yeganeh, S., Ganjali, Y., 2012. Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 19–24.
He, F., Oki, E., 2020. Load balancing model against multiple controller failures in software defined networks. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.
He, K., Khalid, J., Das, S., Gember-Jacobson, A., Prakash, C., Akella, A., Li, L.E., Thottan, M., 2015. Latency in software defined networks: measurements and mitigation techniques. In: ACM SIGMETRICS Performance Evaluation Review. ACM, pp. 435–436.
He, M., Basta, A., Blenk, A., Kellerer, W., 2017. Modeling flow setup time for controller placement in SDN: evaluation for dynamic flows. In: Communications (ICC), 2017 IEEE International Conference on. IEEE, pp. 1–7.

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 7–12.
Hopps, C.E., 2000. Analysis of an Equal-Cost Multi-Path Algorithm, 2000.
Hou, R., Wang, D., Wang, Y., Zhu, Z., 2019. A congestion control methodology with probability routing based on mnl for datacenter network. In: International Conference on Artificial Intelligence and Security. Springer, pp. 343–352.
http://www.sFlow.org/sFlowoverview.pdf, 2018–. (Accessed 9 January 2018).
HU, Y.n., Wang, W.d., Gong, X.y., Que, X.r., Cheng, S.d., 2012. On the placement of controllers in software-defined networks. J. China Univ. Posts Telecommun. 19, 92–171.
Hu, Y., Wendong, W., Gong, X., Que, X., Shiduan, C., 2013. Reliability-aware controller placement for software-defined networks. In: Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, pp. 672–675.
Hu, T., Guo, Z., Lan, J., Zhang, J., Zhao, W., 2017a. Easm: Efficiency-Aware Switch Migration for Balancing Controller Loads in Software-Defined Networking, 2017. arXiv preprint arXiv:1711.08659.
Hu, Y., Luo, T., Beaulieu, N.C., Wang, W., 2017b. An initial load-based green software defined network. Appl. Sci. 7, 459.
Hu, T., Guo, Z., Zhang, J., Lan, J., 2018a. Adaptive slave controller assignment for fault-tolerant control plane in software-defined networking. In: 2018 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.
Hu, T., Yi, P., Guo, Z., Lan, J., Zhang, J., 2018b. Bidirectional matching strategy for multi-controller deployment in distributed software defined networking. IEEE Access 6, 14946–14953.
Hu, T., Yi, P., Guo, Z., Lan, J., Hu, Y., 2019. Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks. Future Generat. Comput. Syst. 95, 681–693.
Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., et al., 2013. B4: experience with a globally-deployed software defined wan. In: ACM SIGCOMM Computer Communication Review. ACM, pp. 3–14.
Jalili, A., Keshtgari, M., Akbari, R., Javidan, R., 2019. Multi criteria analysis of controller placement problem in software defined networks. Comput. Commun. 133, 115–128.
Jalili, A., Keshtgari, M., Akbari, R., 2020. A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach. Soft Comput. 24, 2897–2916.
Jamali, S., Badirzadeh, A., Siapoush, M.S., 2019. On the use of the genetic programming for balanced load distribution in software-defined networks. Digit. Commun. Netw. 5, 288–296.
Javadpour, A., 2020. Providing a way to create balance between reliability and delays in SDN networks by using the appropriate placement of controllers. Wireless Pers. Commun. 110, 1057–1071.
Jia, X., Jiang, Y., Guo, Z., Wu, Z., 2016. Reducing and balancing flow table entries in software-defined networks. In: 2016 IEEE 41st Conference on Local Computer Networks (LCN). IEEE, pp. 575–578.
Jia, X., Li, Q., Jiang, Y., Guo, Z., Sun, J., 2017. A low overhead flow-holding algorithm in software-defined networks. Comput. Network. 124, 170–180.
Jimenez, Y., Cervello-Pastor, C., Garcia, A.J., 2014. On the controller placement for designing a distributed SDN control layer. In: Networking Conference, 2014 IFIP. IEEE, pp. 1–9.
Jimson, E.R., Nisar, K., Hijazi, M.H.A., 2019. The state of the art of software defined networking (SDN) issues in current network architecture and a solution for network management using the SDN. Int. J. Technol. Diffusion (IJTD) 10, 33–48.
Jin, H., Yang, G., Yu, B.y., Yoo, C., 2019. Talon: tenant throughput allocation through traffic load-balancing in virtualized software-defined networks. In: 2019 International Conference on Information Networking (ICOIN). IEEE, pp. 233–238.
Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R., 2009. The nature of data center traffic: measurements & analysis. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. ACM, pp. 202–208.
Karakus, M., Durresi, A., 2017a. Quality of service (QoS) in software defined networking (SDN): a survey. J. Netw. Comput. Appl. 80, 200–218.
Karakus, M., Durresi, A., 2017b. A survey: control plane scalability issues and approaches in software-defined networking (SDN). Comput. Network. 112, 279–293.
Kashiri, N., Tsagarakis, N.G., Van Damme, M., Vanderborght, B., Caldwell, D.G., 2016. Proxy-based sliding mode control of compliant joint manipulators. In: Informatics in Control, Automation and Robotics. Springer, pp. 241–257.
Kaur, S., Kumar, K., Singh, J., Ghumman, N.S., 2015. Round-robin based load balancing in software defined networking. In: Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on. IEEE, pp. 2136–2139.
Kaur, P., Chahal, J.K., Bhandari, A., 2018. Load balancing in software defined networking: a review. Asian J. Comput. Sci. Technol. 7, 1–5.
Khan, S., Gani, A., Wahab, A.W.A., Abdelaziz, A., Bagiwa, M.A., 2016a. Fml: a novel forensics management layer for software defined networks. In: 2016 6th International Conference-Cloud System and Big Data Engineering (Confluence). IEEE, pp. 619–623.
Khan, S., Gani, A., Wahab, A.W.A., Abdelaziz, A., Ko, K., Khan, M.K., Guizani, M., 2016b. Software-defined network forensics: motivation, potential locations, requirements, and challenges. IEEE Network 30, 6–13.
Khan, S., Bagiwa, M.A., Wahab, A.W.A., Gani, A., Abdelaziz, A., 2020. Understanding link fabrication attack in software defined network using formal methods. In: 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT). IEEE, pp. 555–562.
Koerner, M., Kao, O., 2012. Multiple service load-balancing with OpenFlow. In: High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on. IEEE, pp. 210–214.
Khler, F., 2018. Network Virtualization in Multi-Hop Heterogeneous Architecture.
Konglar, K., Somchit, Y., 2018. Load distribution of software-defined networking based on controller performance. In: 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE, pp. 1–6.
Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., et al., 2010. Onix: a distributed control platform for large-scale production networks. In: OSDI, pp. 1–6.
Koponen, T., Casado, M., Gude, N., Stribling, J., 2014. Distributed control platform for large-scale production networks. US Patent 8,830,823.
Kreutz, D., Ramos, F.M., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 55–60.
Kumari, P., Thakur, D., 2017. Load balancing in software defined network. Int. J. Comput. Sci. Eng. 5 (2017), 227–232.
Kurroliya, K., Mohanty, S., Sahoo, B., Kanodia, K., 2020. Minimizing energy consumption in software defined networks. In: 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, pp. 885–890.
Kuniar, M., Pereni, P., Kosti, D., 2015. What you need to know about SDN flow tables. In: International Conference on Passive and Active Network Measurement. Springer, pp. 347–359.
Kyung, Y., Hong, K., Nguyen, T.M., Park, S., Park, J., 2015. A load distribution scheme over multiple controllers for scalable SDN. In: Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on. IEEE, pp. 808–810.
Lam, J.H., Lee, S.G., Lee, H.J., Oktian, Y.E., 2016. Tls channel implementation for onoss east/west-bound communication. In: Electronics, Communications and Networks V. Springer, pp. 397–403.
Latif, Z., Sharif, K., Li, F., Karim, M.M., Biswas, S., Wang, Y., 2020. A comprehensive survey of interface protocols for software defined networks. J. Netw. Comput. Appl. 102563.
Ld, D.B., Krishna, P.V., 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13, 2292–2303.
Leland, R., Hendrickson, B., 1994. An empirical study of static load balancing algorithms. In: Scalable High-Performance Computing Conference, 1994. Proceedings of the, IEEE, pp. 682–685.
Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A., 2012. Logically centralized?: state distribution trade-offs in software defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 1–6.
Li, L., Xu, Q., 2017. Load balancing researches in SDN: a survey. In: Electronics Information and Emergency Communication (ICEIEC), 2017 7th IEEE International Conference on. IEEE, pp. 403–408.
Li, J., Chang, X., Ren, Y., Zhang, Z., Wang, G., 2014. An effective path load balancing mechanism based on SDN. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, pp. 527–533.
Li, W., Meng, W., et al., 2016. A survey on OpenFlow-based software defined networks: security challenges and countermeasures. J. Netw. Comput. Appl. 68, 126–139.
Li, L., Du, N., Liu, H., Zhang, R., Yan, C., 2019. Towards robust controller placement in software-defined networks against links failure. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, pp. 216–223.
Li, H., Lu, H., Fu, X., 2020. An optimal and dynamic elephant flow scheduling for SDN-based data center networks. J. Intell. Fuzzy Syst. 38, 247–255.
Liang, C., Kawashima, R., Matsuo, H., 2014. Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers. In: Computing and Networking (CANDAR), 2014 Second International Symposium on. IEEE, pp. 171–177.
Liao, J., Sun, H., Wang, J., Qi, Q., Li, K., Li, T., 2017a. Density cluster based approach for controller placement problem in large-scale software defined networkings. Comput. Network. 112, 24–35.
Liao, L., Leung, V.C., Lai, C.F., 2017b. Evolutionary algorithms in software defined networks: techniques, applications, and issues. ZTE Commun. 5.
Lin, C.Y., Chen, C., Chang, J.W., Chu, Y.H., 2014. Elephant flow detection in datacenters using OpenFlow-based hierarchical statistics pulling. In: 2014 IEEE Global Communications Conference. IEEE, pp. 2264–2269.
Lin, S.C., Akyildiz, I.F., Wang, P., Luo, M., 2016a. QoS-aware adaptive routing in multi-layer hierarchical software defined networks: a reinforcement learning approach. In: Services Computing (SCC), 2016 IEEE International Conference on. IEEE, pp. 25–33.
Lin, S.C., Wang, P., Luo, M., 2016b. Control traffic balancing in software defined networks. Comput. Network. 106, 260–271.
Lin, T.L., Kuo, C.H., Chang, H.Y., Chang, W.K., Lin, Y.Y., 2016c. A parameterized wildcard method based on SDN for server load balancing. In: Networking and Network Applications (NaNA), 2016 International Conference on. IEEE, pp. 383–386.
Liu, L., Jiang, Y., Shen, G., Li, Q., Lin, D., Li, L., Wang, Y., 2019. An SDN-based hybrid strategy for load balancing in data center networks. In: 2019 IEEE Symposium on Computers and Communications (ISCC). IEEE, pp. 1–6.
Liu, W., Wang, Y., Zhang, J., Liao, H., Liang, Z., Liu, X., 2020. Aamcon: an adaptively distributed SDN controller in data center networks. Front. Comput. Sci. 14, 146–161.
Long, H., Shen, Y., Guo, M., Tang, F., 2013. Laberio: dynamic load-balanced routing in OpenFlow-enabled networks. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE, pp. 290–297.
Lou, K., Yang, Y., Wang, C., 2019. An elephant flow detection method based on machine learning. In: International Conference on Smart Computing and Communication. Springer, pp. 212–220.
Lu, J., Zhang, Z., Hu, T., Yi, P., Lan, J., 2019. A survey of controller placement problem in software-defined networking. IEEE Access 7, 24290–24307.

Ma, Y.W., Chen, J.L., Tsai, Y.H., Cheng, K.H., Hung, W.C., 2017. Load-balancing multiple controllers mechanism for software-defined networking. Wireless Pers. Commun.: Int. J. 94, 3549–3574.
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008a. OpenFlow: enabling innovation in campus networks. Comput. Commun. Rev. 38, 69–74.
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008b. OpenFlow: enabling innovation in campus networks. Comput. Commun. Rev. 38, 69–74.
Mehra, M., Maurya, S., Tiwari, N.K., 2019. Network load balancing in software defined network: a survey. Int. J. Appl. Eng. Res. 14, 245–253.
Michel, O., Keller, E., 2017. SDN in wide-area networks: a survey. In: 2017 Fourth International Conference on Software Defined Systems (SDS). IEEE, pp. 37–42.
Mininet Simulation, 2019. http://Mininet.org/. (Accessed 10 January 2019).
Montazerolghaem, A., Yaghmaee, M.H., 2020. Load-balanced and QoS-aware software-defined internet of things. IEEE Internet of Things J. 7, 3323–3337.
Muluye, W., Abebe, M., Kumar, N.S., 2019. Performance analysis and evaluation of software defined networking distributed controllers in datacentre networks. Int. J. Comput. Syst. Eng. 5, 61–66.
Nayyer, A., Sharma, A.K., Awasthi, L.K., 2019a. Issues in software-defined networking. In: Proceedings of 2nd International Conference on Communication, Computing and Networking. Springer, pp. 989–997.
Nayyer, A., Sharma, A.K., Awasthi, L.K., 2019b. Laman: a supervisor controller based scalable framework for software defined networks. Comput. Network. 159, 125–134.
Neghabi, A.A., Navimipour, N.J., Hosseinzadeh, M., Rezaee, A., 2018. Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature. IEEE Access 6 (2018), 14159–14178.
Ng, E., Cai, Z., Cox, A., 2010. Maestro: A System for Scalable OpenFlow Control. Rice University, Houston, TX, USA. TSEN Maestro-Techn. Rep, TR10-08, 2010.
Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T., 2014. A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. 16, 1617–1634.
Obadia, M., Bouet, M., Rougier, J.L., Iannone, L., 2015. A greedy approach for minimizing SDN control overhead. In: Network Softwarization (NetSoft), 2015 1st IEEE Conference on. IEEE, pp. 1–5.
OpenFlow protocol v1.5. accessed 01 June 2018. https://www.opennetworking.org/wp-content/uploads/2014/10/OpenFlow-switch-v1.5.1.pdf.
Othman, M.O., Okamura, K., 2013. Enhancing control model to ease off centralized control of flow-based SDNs. In: Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual. IEEE, pp. 467–470.
Patel, P., Bansal, D., Yuan, L., Murthy, A., Greenberg, A., Maltz, D.A., Kern, R., Kumar, H., Zikos, M., Wu, H., et al., 2013. Ananta: cloud scale load balancing. Comput. Commun. Rev. 43, 207–218.
Patil, S., 2018. Load balancing approach for finding best path in SDN. In: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, pp. 612–616.
Pfaff, B., Lantz, B., Heller, B., Barker, C., Beckmann, C., et al., 2012. OpenFlow Switch Specification V1. 3.0. Open Netw. Found. Menlo Park, CA, USA, Tech. Rep. ONF TS-006.
Phemius, K., Thales, M.B., 2013. OpenFlow: why latency does matter. In: Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, pp. 680–683.
Phemius, K., Bouet, M., Leguay, J., 2014. Disco: distributed multi-domain SDN controllers. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, pp. 1–4.
Poddar, R., Vishnoi, A., Mann, V., 2015. Haven: holistic load balancing and auto scaling in the cloud. In: Communication Systems and Networks (COMSNETS), 2015 7th International Conference on. IEEE, pp. 1–8.
Polat, H., Polat, O., Cetin, A., 2020. Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models. Sustainability 12, 1035.
Porras, P.A., Cheung, S., Fong, M.W., Skinner, K., Yegneswaran, V., 2015. Securing the software defined network control layer. In: NDSS.
Qazi, Z.A., Tu, C.C., Chiang, L., Miao, R., Sekar, V., Yu, M., 2013. Simple-fying middlebox policy enforcement using SDN. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, pp. 27–38.
Qian, Y., You, W., Qian, K., 2017. Flowvisor vulnerability analysis. In: Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on. IEEE, pp. 867–868.
Qiu, K., Huang, S., Xu, Q., Zhao, J., Wang, X., Secci, S., 2017. Paracon: a parallel control plane for scaling up path computation in SDN. IEEE Trans. Netw. Serv. Manag. 14, 978–990.
Qureshi, K.I., Wang, L., Sun, L., Zhu, C., Shu, L., 2020. A review on design and implementation of software-defined wlans. IEEE Syst. J.
Raghul, S., Subashri, T., Vimal, K., 2017. Literature survey on traffic-based server load balancing using SDN and open flow. In: 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN). IEEE, pp. 1–6.
Ramya, K., Sayeekumar, M., Karthik, G., 2020. Software defined networking based solution in load balancing for media transfer in overlay network. J. Comput. Theor. Nanosci. 17, 43–47.
Research, G.V, 2017. http://www.grandviewresearch.com/industry-analysis/software-defined-networking-SDN-market-analysis. (Accessed 10 January 2017).
Ruelas, A.M., Rothenberg, C.E., 2018. A load balancing method based on artificial neural networks for knowledge-defined data center networking. In: Proceedings of the 10th Latin America Networking Conference, pp. 106–109.
Rupani, K., Punjabi, N., Shamdasani, M., Chaudhari, S., 2020. Dynamic load balancing in software-defined networks using. In: Proceeding of International Conference on Computational Science and Applications: ICCSA 2019. Springer Nature, p. 283.
Sahoo, K.S., Sahoo, B., 2019. Camd: a Switch Migration Based Load Balancing Framework for Software Defined Networks. IET Networks.
Sahoo, K.S., Mishra, P., Tiwary, M., Ramasubbareddy, S., Balusamy, B., Gandomi, A.H., 2019a. Improving end-users utility in software-defined wide area network systems. IEEE Trans. Netw. Serv. Manag. 17, 696–707.
Sahoo, K.S., Puthal, D., Tiwary, M., Usman, M., Sahoo, B., Wen, Z., Sahoo, B.P., Ranjan, R., 2019b. Esmlb: efficient switch migration-based load balancing for multi-controller SDN in iot. IEEE Internet of Things J.
Saldamli, G., Sanjeeva, H., Siddalingaapa, K., Murugesan, R.V., Ertaul, L., 2019. A secure collaborative module on distributed SDN. In: 2019 10th International Conference on Information and Communication Systems (ICICS). IEEE, pp. 65–70.
Salman, M.A., Bertelle, C., Sanlaville, E., 2014. The behavior of load balancing strategies with regard to the network structure in distributed computing systems. In: Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on. IEEE, pp. 432–439.
Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. SDN security: a survey. In: Future Networks and Services (SDN4FNS), 2013. IEEE SDN For, IEEE, pp. 1–7.
Selvi, H., Gr, G., Alagz, F., 2016. Cooperative load balancing for hierarchical SDN controllers. In: High Performance Switching and Routing (HPSR), 2016 IEEE 17th International Conference on. IEEE, pp. 100–105.
Semong, T., Maupong, T., Anokye, S., Kehulakae, K., Dimakatso, S., Boipelo, G., Sarefo, S., 2020. Intelligent load balancing techniques in software defined networks: a survey. Electronics 9, 1091.
eremet, I., auevi, S., 2020. Advances of configuring quality of service (QoS) in software defined networks (SDN) by using meter table. In: 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH). IEEE, pp. 1–5.
Shabtay, L., 2010. Dynamic load balancer. US Patent 7,739,398.
Shaghaghi, A., Kaafar, M.A., Buyya, R., Jha, S., 2020. Software-defined network (SDN) data plane security: issues, solutions, and future directions. In: Handbook of Computer Networks and Cyber Security. Springer, pp. 341–387.
Shah, S.A., Faiz, J., Farooq, M., Shafi, A., Mehdi, S.A., 2013. An architectural evaluation of SDN controllers. In: Communications (ICC), 2013 IEEE International Conference on. IEEE, pp. 3504–3508.
Shang, Z., Chen, W., Ma, Q., Wu, B., 2013. Design and implementation of server cluster dynamic load balancing based on OpenFlow. In: Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on. IEEE, pp. 691–697.
Shang, F., Mao, L., Gong, W., 2018. Service-aware adaptive link load balancing mechanism for software-defined networking. Future Generat. Comput. Syst. 81, 452–464.
Sherwood, R., Gibb, G., Yap, K.K., Appenzeller, G., Casado, M., McKeown, N., Parulkar, G., 2009. Flowvisor: a network virtualization layer. OpenFlow switch consortium. Tech. Rep. 1, 132.
Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.Y., Kazemian, P., Kobayashi, M., Naous, J., et al., 2010a. Carving research slices out of your production networks with OpenFlow. Comput. Commun. Rev. 40, 129–130.
Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.Y., Kazemian, P., Kobayashi, M., Naous, J., et al., 2010b. Carving research slices out of your production networks with OpenFlow. Comput. Commun. Rev. 40, 129–130.
Shi, X., Li, Y., Xie, H., Yang, T., Zhang, L., Liu, P., Zhang, H., Liang, Z., 2020. An OpenFlow-Based load balancing strategy in SDN. CMC-Computers Materials & Continua, 62, pp. 385–398.
Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 413–424.
Shirmarz, A., Ghaffari, A., 2020. Performance issues and solutions in SDN-based data center: a survey. J. Supercomput. 1–49.
Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A.V., Imran, M., 2016. Security in software-defined networking: threats and countermeasures. Mobile Network. Appl. 21, 764–776.
Singh, M.P., Bhandari, A., 2020. New-flow based DDoS attacks in SDN: taxonomy, rationales, and research challenges. Comput. Commun. 2020.
Spalla, E.S., Mafioletti, D.R., Liberato, A.B., Ewald, G., Rothenberg, C.E., Camargos, L., Villaca, R.S., Martinello, M., 2016. Ar2c2: actively replicated controllers for SDN resilient control plane. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 189–196.
Srinidhi, N., Kumar, S.D., Venugopal, K., 2019. Network optimizations in the internet of things: a review. Eng. Sci. Technol. Int. J. 22, 1–21.
Srivastava, V., Pandey, R.S., 2020. A dominance of the channel capacity in load balancing of software defined network. Wireless Pers. Commun. 1–15.
Stribling, J., Sovran, Y., Zhang, I., Pretzer, X., Li, J., Kaashoek, M.F., Morris, R., 2009. Flexible, wide-area storage for distributed systems with wheelfs. In: NSDI, pp. 43–58.
Sun, G., Xu, Z., Yu, H., Chen, X., Chang, V., Vasilakos, A.V., 2019a. Low-latency and resource-efficient service function chaining orchestration in network function virtualization. IEEE Internet of Things J. 2019.
Sun, P., Li, J., Guo, Z., Xu, Y., Lan, J., Hu, Y., 2019b. Sinet: enabling scalable network routing with deep reinforcement learning on partial nodes. In: Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos, pp. 88–89.
Sun, P., Guo, Z., Wang, G., Lan, J., Hu, Y., 2020a. Marvel: Enabling Controller Load Balancing in Software-Defined Networks with Multi-Agent Reinforcement Learning. Computer Networks, 107230.
Sun, P., Lan, J., Guo, Z., Xu, Y., Hu, Y., 2020b. Improving the scalability of deep reinforcement learning-based routing with control on partial nodes. In: ICASSP

2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 3557–3561.
Tadros, C.N., Mokhtar, B., Rizk, M.R., 2018. Logically centralized-physically distributed software defined network controller architecture. In: 2018 IEEE Global Conference on Internet of Things (GCIoT). IEEE, pp. 1–5.
Tam, A.S.W., Xi, K., Chao, H.J., 2011. Use of devolved controllers in data center networks. In: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, pp. 596–601.
Tao, F., Jun, B., Ke, W., 2015. Allocation and scheduling of network resource for multiple control applications in SDN. China Commun. 12, 85–95.
Thi, M.T., Huynh, T., Hasegawa, M., Hwang, W.J., 2017. A rate allocation framework for multi-class services in software-defined networks. J. Netw. Syst. Manag. 25, 1–20.
Tootoonchian, A., Ganjali, Y., 2010. Hyperflow: a distributed control plane for OpenFlow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, 33.
Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R., 2012. On controller performance in software-defined networks. Hot-ICE, 12, pp. 1–6.
Tuncer, D., Charalambides, M., Clayman, S., Pavlou, G., 2015. Adaptive resource management and control in software defined networks. IEEE Trans. Netw. Serv. Manag. 12, 18–33.
Uppal, H., Brandon, D., 2010. OpenFlow Based Load Balancing. CSE561: Networking Project Report. University of Washington, 2010.
Wang, Y., Tao, X., He, Q., Kuang, Y., 2016. A dynamic load balancing method of cloud-center based on SDN. China Commun. 13, 130–137.
Wang, C., Zhang, G., Xu, H., Chen, H., 2016a. An aco-based link load-balancing algorithm in SDN. In: 2016 7th International Conference on Cloud Computing and Big Data (CCBD). IEEE, pp. 214–218.
Wang, W., Sun, Y., Salamatian, K., Li, Z., 2016b. Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters. IEEE Trans. Netw. Serv. Manag. 13, 5–18.
Wang, C., Zhang, G., Chen, H., Xu, H., 2017. An aco-based elephant and mice flow scheduling system in SDN. In: 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA). IEEE, pp. 859–863.
Win, M.T.Z., Ishibashi, Y., Mya, K.T., 2019. QoS-aware traffic engineering in software defined networks. In: 2019 25th Asia-Pacific Conference on Communications (APCC). IEEE, pp. 171–176.
Xiao, P., Qu, W., Qi, H., Li, Z., Xu, Y., 2014. The SDN controller placement problem for wan. In: Communications in China (ICCC), 2014 IEEE/CIC International Conference on. IEEE, pp. 220–224.
Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Wang, C., Liu, Y., 2018. A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges. IEEE Commun. Surv. Tutor. 21, 393–430.
Xu, Y., Liu, Z., Zhang, Z., Chao, H.J., 2009. An ultra high throughput and memory efficient pipeline architecture for multi-match packet classification without tcams. In: Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 189–198.
Xu, Y., Liu, Z., Zhang, Z., Chao, H.J., 2013. High-throughput and memory-efficient multimatch packet classification based on distributed and pipelined hash tables. IEEE/ACM Trans. Netw. 22, 982–995.
Xu, Y., Cello, M., Wang, I.C., Walid, A., Wilfong, G., Wen, C.H.P., Marchese, M., Chao, H. J., 2019. Dynamic switch migration in distributed software-defined networks to achieve controller load balance. IEEE J. Sel. Area. Commun. 37, 515–529.
Xue, H., Kim, K.T., Youn, H.Y., 2019. Dynamic load balancing of software-defined networking based on genetic-ant colony optimization. Sensors 19, 311.
Yan, Z., Zhang, P., Vasilakos, A.V., 2016. A security and trust framework for virtualized networks and software-defined networking. Secur. Commun. Network. 9, 3059–3069.
Yang, M., Li, Y., Jin, D., Su, L., Zeng, L., 2014. Opportunistic spectrum sharing in software defined wireless network. J. Syst. Eng. Electron. 25, 934–941.
Yao, G., Bi, J., Guo, L., 2013. On the cascading failures of multi-controllers in software defined networks. In: Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, pp. 1–2.
Yazici, V., Sunay, M.O., Ercan, A.O., 2014. Controlling a Software-Defined Network via Distributed Controllers, 2014. arXiv preprint arXiv:1401.7651.
Yi, B., Wang, X., Li, K., Huang, M., et al., 2018. A Comprehensive Survey of Network Function Virtualization. Computer Networks, 2018.
Yu, J., Wang, Y., Pei, K., Zhang, S., Li, J., 2016. A load balancing mechanism for multiple SDN controllers based on load informing strategy. In: Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific. IEEE, pp. 1–4.
Yu, C., Lan, J., Guo, Z., Hu, Y., 2018. Drom: optimizing the routing in software-defined networks with deep reinforcement learning. IEEE Access 6, 64533–64539.
Zarek, A., Ganjali, Y., Lie, D., 2012. OpenFlow Timeouts Demystified. Univ. of Toronto, Toronto, Ontario, Canada.
Zehra, U., Shah, M.A., 2017. A survey on resource allocation in software defined networks (SDN). In: Automation and Computing (ICAC), 2017 23rd International Conference on. IEEE, pp. 1–6.
Zeng, X., Wang, D., Han, S., Yao, W., Wang, Z., Chen, R., 2019. An effective load balance using link bandwidth for SDN-Based data centers. In: International Conference on Artificial Intelligence and Security. Springer, pp. 256–265.
Zhang, H., Guo, X., 2014. SDN-based load balancing strategy for server cluster. In: Cloud Computing and Intelligence Systems (CCIS), 2014 IEEE 3rd International Conference on. IEEE, pp. 662–667.
Zhang, L., Shou, G., Hu, Y., Guo, Z., 2013. Deployment of intrusion prevention system based on software defined networking. In: Communication Technology (ICCT), 2013 15th IEEE International Conference on. IEEE, pp. 26–31.
Zhang, Y., Cui, L., Wang, W., Zhang, Y., 2017. A survey on software defined networking with multiple controllers. J. Netw. Comput. Appl. 2017.
Zhang, J., Yu, F.R., Wang, S., Huang, T., Liu, Z., Liu, Y., 2018. Load balancing in data center networks: a survey. IEEE Commun. Surv. Tutor. 20, 2324–2352.
Zhang, J., Ye, M., Guo, Z., Yen, C., Chao, H.J., 2020. Cfr-rl: traffic engineering with reinforcement learning in SDN. IEEE J. Sel. Area. Commun. 11.
Zhao, Z., Wu, B., 2017. Scalable SDN architecture with distributed placement of controllers for wan. Concurrency Comput. Pract. Ex. 29, e4030.
Zhao, J., Hu, Z., Xiong, B., Yang, L., Li, K., 2020a. Modeling and optimization of packet forwarding performance in software-defined wan. Future Generat. Comput. Syst. 106, 412–425.
Zhao, Y., Wang, X., Zhang, C., He, Q., Huang, M., 2020b. Power optimization with less state transition for green software defined networking. Future Generat. Comput. Syst.
Zhong, H., Fang, Y., Cui, J., 2017a. Lbbsrt: an efficient SDN load balancing scheme based on server response time. Future Generat. Comput. Syst. 68, 183–190.
Zhong, H., Sheng, J., Cui, J., Xu, Y., 2017b. SCPLBS: a smart cooperative platform for load balancing and security on SDN distributed controllers. In: Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on. IEEE, pp. 1–6.
Zhong, H., Fang, Y., Cui, J., 2018. Reprint of lbbsrt: an efficient SDN load balancing scheme based on server response time. Future Generat. Comput. Syst. 80, 409–416.
Zhong, H., Sheng, J., Xu, Y., Cui, J., 2019. Scplbs: a smart cooperative platform for load balancing and security on SDN distributed controllers. Peer-to-Peer Netw. Appl. 12, 440–451.
Zhou, W., Yang, S., Fang, J., Niu, X., Song, H., 2010. Vmctune: a load balancing scheme for virtual machine cluster using dynamic resource allocation. In: Grid and Cooperative Computing (GCC), 2010 9th International Conference on. IEEE, pp. 81–86.
Zhou, Y., Zhu, M., Xiao, L., Ruan, L., Duan, W., Li, D., Liu, R., Zhu, M., 2014b. A load balancing strategy of SDN controller based on distributed decision. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, pp. 851–856.
Zhou, W., Li, L., Luo, M., Chou, W., 2014. Rest api design patterns for SDN northbound api. In: Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on. IEEE, pp. 358–365.
Zhou, Y., Wang, Y., Yu, J., Ba, J., Zhang, S., 2017. Load balancing for multiple controllers in SDN based on switches group. In: Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific. IEEE, pp. 227–230.
Zou, Y., Tian, Y., Guo, S., Wu, Y., 2017. Active synchronization of multi-domain controllers in software-defined networks. Concurrency Comput. Pract. Ex. 29, e3979.
https://opennetworking.org/wp-content/uploads/2014/10/%7BOpenFlow%7D-switch-v1.5.1.pdf. (Accessed 1 June 2018).

Mosab Hamdan received a B.Sc. degree in Computer and Electronic System Engineering from the University of Science and Technology (Sudan) in 2010. He received an M.Sc. in Computer Architecture and Networking from the University of Khartoum (Sudan) in 2014. He is currently pursuing his Ph.D. degree in the School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. His current research interests are software-defined networking (SDN), load balancing, network traffic classification, and future network.

Entisar Hassan received the B.Sc. degree in Electronics engineering from the University of al-Jazirah (Sudan) in 2006. She received the M.Sc. in telecommunications and information systems from the University of Khartoum (Sudan) in 2009. She is currently pursuing his Ph.D. degree in the School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. Her current research interests are Software Defined Networking (SDN), Deep learning, network traffic classification, and future network.

Ahmed Abdelaziz received the M.Sc. degree in computer science and the Ph.D. degree in information technology from the Universiti Malaya (UM), Malaysia, in 2007 and 2017, respectively. He has been working on ONOS and Open Stack, since October 2015, during the Ph.D. degree research project. In the Ph.D. degree research, he proposed a novel service-based load balancing technique to use in the cloud using SDN and OpenStack. He is currently a full-time Assistant Professor with Future University (FU), Sudan. He published a number of ISI index articles in the areas of SDN, OpenFlow, and network virtualization. He has been involved in the Centre for Mobile Cloud Computing Research (C4MCCR) Projects funded by the Malaysian Ministry of Higher Education. His areas of interest include SDN/NFV technology, OpenStack, and network virtualization.

Abdallah Elhigazi received a B.Sc. degree in Computer Science from the University of Science and Technology (Sudan) in 2004. He received his M.Sc. degree in Computer Science from the University of Gezira (Sudan) in 2007. He received a Ph.D. degree in 2020 in the School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia. His current research interests in wireless network security, machine learning artificial intelligence, information assurance, and security research.

Bushra Mohammed received the B.Sc. and M.Sc. degree in Computer Engineering and Networks from the University of Gezira, Sudan and the PhD degree in Electrical Engineering from Universiti Teknologi Malaysia, in 2020. He is a lecturer at the faculty of Computer and Statistics Studies, University of Kordofan, Sudan. His research interests include computer architecture, network traffic classification and control, Artificial Intelligence and optimization techniques.

Suleman Khan received the Ph.D. degree (Distinction) from the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia, in 2017. He was a faculty member with the School of Information Technology, Monash University Malaysia (June 17-March 19). Currently, he is a faculty member in the Department of Computer and Information Sciences, Northumbria University, Newcastle, UK. He has published more than 50 high-impact research articles in reputed international journals and conferences. His research areas include, but are not limited to, network forensics, software-defined networks, the Internet-of-things, cloud computing, and vehicular communications.

Athanasios V. Vasilakos is a distinguished full professor with the School of Electrical and Data Engineering, University Technology Sydney, Australia (email:th.vasilakos@gmail.com). He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Cybernetics, the IEEE Transactions on Nanobioscience, the IEEE Transactions on Information Technology in Biomedicine, the ACM Transactions on Autonomous and Adaptive Systems, and the IEEE Journal on Selected Areas in Communications. He is Web of Science 2017, 2018, 2019, 2020 highly cited researcher. He is also general chair of the european alliances for innovation (http://www.eai.eu).

M. N. Marsono is an associate professor in Electronic and Computer Engineering, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. He obtained his Ph.D. in Electrical and Computer Engineering from the University of Victoria BC Canada in 2007, M.Eng. in Electrical Engineering from Universiti Teknologi Malaysia in 2001, and B.Eng. in Computer Engineering from Universiti Teknologi Malaysia in 1999. His research focuses on specialized hardware architecture and network algorithmics for high-throughput packet and flow processing. He works on dynamically reconfigurable platforms for middlebox, fog and edge computing, software-defined networking, and teletraffic engineering. He also works in domain-specific reconfigurable computing research, focusing on multicore/manycore system-on-chip, network-on-chip, design space exploration, mapping, and prototyping of the homogeneous and heterogeneous manycore SoC.