SN Computer Science (2020) 1:268
https://doi.org/10.1007/s42979-020-00288-8

SURVEY ARTICLE

A Taxonomy of Load Balancing Mechanisms in Centralized and Distributed SDN Architectures

Farah Chahlaoui[1] · Hamza Dahmouni[1]

Received: 24 May 2020 / Accepted: 5 August 2020 / Published online: 14 August 2020

Abstract
Recent Research have shown that software-defined networking (SDN) drastically improves network resource utilization, manages today's complex networks and reduces cost. The rapid development of technology and the explosively growing information services as well as the increasing number of users, have made the load distribution a huge issue that could eventually impact the network's performance and its quality of service. The need for SDN comes from the load-balancing problematic which could reduce the availability of network applications, resource utilization and QoS indicators such as latency, throughput and response time. In this paper, we analyze the impact of software-defined networks design and the architecture of its control plane on the load-balancing methods efficiency. We present a detailed analysis of some load-balancing algorithms and metrics in centralized and distributed SDN architectures. We also introduce some of the load-balancing mechanisms applied in 5G networks in particular.

Keywords Load balancing · Software defined networks · OpenFlow · SDN architecture · 5G · QoS

Introduction

Software defined networking [1, 2] provide a centralized network and trafic management. The network administrators can program forwarding decision calculation by using the OpenFlow protocol [3] in a centralized manner. Its general view of the network, its programmability and cooperativeness enables SDN with a better management of the traffic flows in the network taking into consideration their characteristics and quality of service requirements. SDN also provides the network with a better flexibility and reactivity to the network's unexpected changes. Load balancing issue in SDN is still not thoroughly studied and more investigations should be done on this area of the field. The ultimate objective is to reactively indicate to the incoming trafic the best transmission paths in the network with the highest quality of service possible.

✉ Farah Chahlaoui
chahlaoui@inpt.ac.ma

Hamza Dahmouni
dahmouni@inpt.ac.ma

[1] Institut National des Postes et Télécommunications, INPT, 2, av. Allal El Fassi, Madinat Al Irfane, 10000 Rabat, Morocco

Compared to traditional networks, software-defined networks (SDN) paradigm [4] have many advantages including the efficiency and simplicity of network management and security policies application. SDN overcomes the traditional networks short-comings that includes limited and inaccurate view of the network causing their inability to achieve end-to-end quality of service and optimized load balancing.

The architecture of the controller has a great impact on the performance of SDN. Numerous controller architectures have been proposed in the body of research, among them the centralized controller, the distributed but logically centralized controller and the completely distributed controller. Each architecture has its advantages and drawbacks. The centralized single controller for example showed scalability limitations. The distributed deployment based on multiple controllers, however, improves the scalability of the control plane but with a high risk of uneven load distribution while some are overloaded other controllers might be unused. It is compulsory to use the adequate load balancing algorithm for each architecture to establish an optimal load management scheme.

The tremendous demand of internet traffic with the 5G deployments is inciting mobile operators and researchers to provide solutions in terms of access capacity and network transport resources to cope with this explosive growth. 5G networks offer different services over network slices with disparate quality of service requirements. 5G networks infrastructures aims to simultaneously handle different applications, highly demanding and low bit rate traffic and reduce congestion. Many solutions are proposed to tackle the rise of dense heterogeneous networks and large traffic prerequisites such as distributed mobility management (DDM), spectrum sharing that ensures the necessary QoS for users and alleviates capacity limitations.

The massive connectivity that will result from fifth-generation network will have a major impact on the mobile network architecture. The increase in the traffic volume, the indoor and small cells and the number of connected devices (internet of thing and machine-to-machine communications) led the researchers to consider multiple technologies to ensure high-speed data and low latency. Among these technologies, load balancing and network traffic offloading are essential ones.

The goal of 5G networks is also to achieve flexibility and reduce dependence on proprietary technologies and dedicated hardware to reduce CAPEX and OPEX costs. Based on the forecast, the mobile network operators and the service providers are all switching to virtualized and Software defined networks based technologies and generic platforms. In this paper, we will examine load balancing methods proposed for SDN-based networks as an effort to anticipate their usage in future mobile network architecture.

In this paper, we focus on load-balancing algorithms based on control plane's design. The remainder of this paper is organized as follows. We start by giving a lightweight definition of the load-balancing mechanism in the section. In the next section, we present the SDN controllers architectures and we study the particularity of both centralized and distributed control plane architectures. The next section tackles the load balancing in SDN and introduces its objectives and metrics. The following section gives a taxonomy of existing load-balancing methods classified according to the control plane design of the different SDN architectures. In the next section, the importance of SDN paradigm for 5G core network alongside with load balancing is outlined. Finally, we conclude the paper citing some of the limitations of the SDN paradigm, outline the challenges that still need to be addressed and identify several important research directions.

Load Balancing Mechanism

Load balancing is the method used to distribute and allocate the incoming clients, requests and tasks efficiently to the available resources in the network [5]. Load balancing objective is to optimize resources utilization in order to avoid congestion and data loss. It could be implemented on hardware as well as software. The load balancer should also have a backup to prevent it from being a single point of failure.

Load balancing algorithms could be categorized as static methods or dynamic methods [6]. The first suits only stable environment since they lack flexibility and adaptability to dynamic changes in the networks. In each of the methods the forwarding rules are already installed in the network managing node and due to the unpredictable aspect of the users' behaviour the static methods are doomed to fail. The dynamic load balancing, however, is more efficient because the transmitted data is distributed according to the network's current status and to the rules programmed in the network in dynamic matter.

Load-Balancing Algorithms

We list in this section some of the load balancing algorithms currently used in today's networks with a reference to their input and short-comings.

Random load-balancing algorithm: this algorithm is a static stateless load balancing strategy. It randomly distributes the workload on resources randomly. This strategy, an incoming traffic is assigned to a resource i with the probability $1/N$, where N is the number of servers. During this process, the controller has no information on the network's state not the follow-up on the traffic. This method was proved to be the worse load balancing strategy with the highest response time [7].

Round Robin load-balancing algorithm: this algorithm routes the incoming traffic evenly to the resources based on Round Robin scheduling algorithm [8]. This means that the dispatcher chooses the destination in series, and sends the traffic to the first resource if the resource N is reached, where N is the number of the resources. The round-robin algorithm uses a scheduling that forwards each incoming packet or request to the following server in it's list until completing the cycle. All the resources (i.e. servers) have equal priority without regard of the number of incoming packets and the response time of the servers. This explains the huge waiting time of this method which impacts the over-all latency.

Weighted Round Robin load-balancing algorithm: This algorithm is an upgrade of Round Robin. Each resource is assigned a weight in the beginning, which defines the workload that will be assigned to each of them and the number of entries each will have in the Round Robin circular queue [9]. It is a static algorithm and does not take into consideration the evolution of the traffic in the network nor the unexpected changes that occurs. In the term of end-to-end latency and response time, this native algorithm is not optimized. It is possible that the incoming traffic is deflected to the farthest server going through a larger number of switches.

Least connection load-balancing algorithm: is a dynamic method of load balancing, the load balancer oversees the number of the connections of each server. The count may increase or decrease depending on the addition of a connection or its release in case of a time-out. The load-balancer chooses the server with the lowest number of current active connections. The least connection method dynamically distribute connections based on real-time server's performance but is unaware of the incoming traffic size which impacts the overall network's performance. This algorithm is suitable to ensure load balancing in a network composed of servers with similar capacity [7, 8].

First come first serve load-balancing algorithm: in this case the load balancer maintains a job queue in which the incoming requests wait for their turn to be assigned to the network's resources. Although FCFS is presented as a simple and fast load-balancing algorithm, it also was proven that its results are overall poor in terms of response time and latency since it doesn't take into account the task size nor its priority or nature [10].

Load Balancing in Traditional Networks

Traditional load balancers policies are static, they are implemented in advance and they lack flexibility leading to a bad performance in case of emergencies. In [11], the authors use for load-balancing MPLS's flexible LSP provisioning. In [12], ECMP is used. It is a mechanism that spreads traffic on paths with equally minimum costs. Both these methods were proved to be highly complex and with high cost. For the Round robin [13] based load-balancing mechanism, this traffic forwarding method works exclusively on paths with equal costs but can cause an unbalanced usage of resources. In [14], Hashing approaches were presented but they are stateless which implies that they are unable to dynamically apply traffic management and load balancing.

In [15], three load balancing methods were presented and evaluated: topology-based static load-balancing algorithm, resource-based static load-balancing algorithm and dynamic load balancing. In both TSLB and RSLB, the load balancing methods are static. Only the criteria of route selection differs from a method to the other. In the first method, the route with the shortest paths from ingress to egress node is chosen with the condition that its available resources and capacity are greater than the incoming traffic bandwidth request. In the second method, the selected route has the nearest free-capacity to the required bandwidth, as to leave the paths with higher capacity for the other more demanding arriving requests. Both methods don't take into account the unpredictable aspect of internet. The networks resources utilization is also low, could be unreasonably used and may cause data loss.

The third method is DLB which combines network's topology and traffic bandwidth request while selecting the routing path. At low load, the traffic flows are sent on low and high capacity routers. At heavy load, the small traffic flows are routed in a manner that reserves high capacity path to large traffic flows. But this method, in spite of the dynamic aspect of load balancing, presents with a high complexity and remains useless in a network with a large amount of parallel routes, nor when the traffic is under priority conditions.

Load Balancing in SDN and OpenFlow Protocol

Among the main features of SDN and openflow networks is their capacity to have a global, up to date, view on the entire network [4, 16,17]. On the other hand, the created dataflow due to control plane signalling and message exchanging may overwhelm the controller and worsen delay and the integrity of transmitted information. This need motivates the work on load balancing technology in SDN which will help the network's elements to prevent eventual congestion and service denial when the network is overflooded. When performing load balancing [18], the SDN controller purpose is to improve resource utilization through program codes that ensures an efficient load management in the network. This improves automatically the availability of the system services and applications. The throughput of the network is improved alongside with its components response time and improves the user's experience.

Openflow protocol, which is presented as a standardized interface that ensure the management of flow tables which store traffic entries, is a key factor in SDN load balancing. Each flow entry is composed of match field, counters and a set of associated instructions and actions. With the use of openflow protocol [19], controllers are able control the flow entries in the flow table and also probe switch statistics. According to the policies and rules installed by the controller an OpenFlow switch [20] acts as a load balancer.

Flow-based switches are flow management architectures that with the use of network policies enforced by the controller handle the network's incoming flow. DIFANE which is a scheduling mechanism is proposed in [21]. It distributes the rules to the switches in order for them to control and manage large networks that have a high number of rules. To this end wildcard rules are used. They implement the defined rules on the corresponding packet that matches it in the data plane. DIFANE is also scalable as it takes into account the networks changes and the mobility of the network components and its hosts. The result is obtaining a network with a high throughput, a small delay and a quick recovery from service drops. In [22], the authors present a network with a centralized flow management solution based on NOX controller. NOX network enables the manipulation of the switches as advised by the decisions taken on the management level. The switch applies the actions defined and that corresponds to the flow entry matching the incoming packet.

Classification Of SDN Controller Architectures

In the remainder of this discussion, we will discuss the SDN networks design.

Centralized Architecture

The centralized SDN architecture have a single controller, multiple switches and servers.The controller manages the traffic allocation the whole network. According to the load balancing strategy adopted, the controller collects the load information of the network and adjusts in real-time and dynamically adjust the load of the network's servers and links. Unnecessary congestion can be avoided by allocating traffic to under-utilized servers. Link load imbalance which increases queuing time and transmission delay may also be corrected with the suitable path load balancing decision [23, 24]. The centralized control architecture [25, 26, 27], in software-defined networks provide a global view of the network that allows a better network performance and resources utilization. Ethane uses a single controller to manage its entire network. However, this architecture has limitations related to scalability, capacity and single point failure [28, 29]. With the growing amount of flows the control can be the bottleneck of its network.

Distributed Architecture

In large networks like Data Centers and Service Provider Networks with thousands of switches it is risky to have only one controller because of the risk to have a single point of failure. Researches on SDN network design have showed some limiting aspects of using of a physically centralized controllers in terms of responsiveness reliability, and scalability. Load-balancing [30] in an SDN network requires a distributed control plane implementation for scalability matter, but also requires a real-time view of the network to optimize the objective function of its algorithm. As a solution, the distributed SDN control was proposed.

The multiple-control plane can be categorized into two different implementation methods: logically centralized or logically distributed. The first is the physically distributed control plane composed of a multiple controllers but they all operate on a logically centralized control plane, which is an architecture where the controllers synchronize their local views of the network and advertise their state to each other. This allow them to construct a solid real-time view of the network and make optimal decisions. However the frequent information exchanges [31, 32] may lead to network overload which impacts the time efficiency of load management decisions including load balancing decisions. Therefore some studies propose the completely distributed controller which is an architecture that composes of multiple controllers and each one controls a different slice of the network. In this method, the load-balancing process is called distributed decision where the network controllers have the same authorities and duties and split the load equally. Each controller has a view of its domain and make the required load balancing decisions locally which reduces the overhead of the communications [33].

Load Balancing Approach in SDN

Load Balancing Objectives

In addition to ensuring a better load distribution among the available network's resources, load-balancing also aims to guarantee the following:

Scalability: In large networks, with the increase of the system's size the scalability is harder to achieve. Scalability is a complex issue in both traditional networks and SDN. In SDN, scalability can be split into node and controller's scalability. It aims to enhance the network's capacity to handle the increasing load [34, 35, 36]. In [37], the authors observed that if the network's quantity of end users and components increases, the SDN controller will face performance issues increasing the delay of the update period of the switch's forwarding information [38]. In [39], the authors propose an elastic distributed controller architecture where the controller pool grows according to load increase. This mechanism allow the node migration from one controller to another less loaded dynamically and achieve load-balancing along with scalability. Moreover, the escalation of the flows quantity and bandwidth, incoming traffic is likely to cause a congestion controller level. Solutions were proposed to allow the controllers to have the same view of network in [40]. The concept is to share the network's state over multiple controllers. With load balancing algorithms the network will be provided with a better scalability since the network resources should be operational and available as to address the increasing demand of services and users.

Resilience: Software defined networks is expected to achieve resilient communications in situations where the network is under attack, in case of components failure and when it faces increasing load [41, 42]. To ensure continuous services, SDN networks must be reliable and must have back-up components and strategies to recover quickly [43]. Load balancing is one of the key mechanisms that guarantee network's resilience by providing reactive and preventive load management policies, redundant and diverse paths [44, 45] between network nodes and the use of multiple controllers. The use of these mechanisms reduces the networks packets loss and latency. The load balancing aspect of resilience is discussed in many studies [46–49].

Resource utilization: Load-balancing solves network congestion and servers overload by sending traffic and network requests to the network resources such as servers and links in a manner that doesn't cause congestion to improve network performance. Load balancing aims to establish an efficient and fair resources utilization [50] to avoid overwhelming an element on the expense of an under-utilized lightly loaded one. The degree of resource utilization such as links, switches, controllers and memory utilization is therefore maximized when using a proper load balancing algorithm. When load is evenly distributed among multiple available resources the overall network performance is enhanced [51] , the delay and response time are reduced, and the throughput is increased. In [52], the authors propose a framework that allows adaptive resource management operations involving short timescale reconfiguration of the available resources. This method was evaluated using real traffic measures and the results show a significant optimization of link utilization and energy consumption.

Quality of service: The main load balancing's objective is provide network's end-to-end quality of service. Hence improve the system's efficiency and performance overall. The quality of service [53] aims to achieve a better user experience which is met by avoiding significant system delay, achieving optimal resource utilization, maximizing throughput and minimizing response time. For each network element, the response time [54] and the delay increase with load increase. Load balancing is thereby required to enable workload even distribution, avoid bottlenecks and resources mismanagement [55]. To measure the impact of load balancing on quality of service we establish later on the paper some metrics such as latency, response time, packet loss rate and throughput.

Load Balancing Metrics

To evaluate the success of a load balancing algorithm, numerous parameters could be taken into account. Papers use a variety of metrics to state the advantages and the limitations of the different existing methods. Bellow a synthesis of some of the most used parameters regarding the four objectives mentioned in the previous section.

1. Scalability: In wide area networks as the size of the network rises, a centralized architecture cannot meet the need concerning scalability then distributed architectures where multiple domains and multiple controllers are used.The scalability of a such system is measured by its ability to maintain its productivity when the network's scale change. One of the used metric is based on the productivity [56]. For a network with N hosts, it is defined by the function $F(N) = \phi(N) \times T(N)/C(N)$ where $\phi(N)$ is the throughput of the control plane; $T(N)$ is the average response time of network requests, $C(N)$ is the deployment cost of the control plane [57, 39].
2. Resilience: For this objective, the controller failures and network disruption are some of the main metrics. In case of a controller failure it is necessary to ensure the reassignment of its nodes to a backup controller. Network disruption could also be caused by link or node failure. Load balancing ensures fair and even node assignment to controllers in these cases in order to avoid congestion and load imbalance between controllers. In [58], the authors propose π_{fail} as a metric to govern controller failures which considers the distance to the backup controllers.
3. Resource utilization: In order to measure the impact of load-balancing on resource utilization optimization several metrics have been proposed: Bandwidth utilization ratio: This metric evaluates the network's transmission and reflects the load condition of the links.The SDN controller calculates the link's bandwidth ratio based on the cumulative transmitted bytes at the corresponding switches ports of two successive periods [59]. The difference is the transmitted data bytes or the band utilization during this period. We, then, divide the latter with the maximum bandwidth ratio to get the bandwidth utilization ratio. A more general aspect to it is the Resource Utilization metric which is the percentage of the resources usage [60]. Overhead: It is the excessive memory, bandwidth, or other resources used to carry communication informations, flow statistics or synchronization data [61, 62]. Forwarding entries: The lower the number of forwarding entries, which decides the packet management rules, is the more effective is the usage of memory resources [63].
4. Quality of service: Numerous metrics are used to measure the QoS of a network. The most significant for load balancing in the SDN are described below:

Latency: Transmission latency is the time taken by a packet to be transmitted from the source to the destination throughout a network. It is related to the time spent by the switch to forward the incoming data and depends on the networks congestion status and the size of the transmitted data. To calculate the latency of a path we calculate the sum of the the transmission delay of each link of this path. It reflects the degree of the network's congestion [64, 65]. Completion time is also used as a metric to measure the execution time of a load balancing algorithm. It includes the migration time of the switches and routing time in the network [66, 67].

Response time: It is a metric that indicates the networks ability to meet QoS requirements in terms of availability, reactivity and delay. It is defined as the interval of time between request acquisition and acceptance until the request response [68]. When the network is congested and suffers from poor load balancing, the packet generation rate at times exceeds the capacity network's resources , causing queuing response time to increase [69, 70]. In [71], the server's response time is the key parameter in the used load balancing algorithm. The response time based load-balancing was also discussed in [72] where the concept of controller's busyness was introduced as the difference between the target response time and the current average response time. Evaluating the response time of a network allows to effectively measure and prove the effectiveness of a load-balancing scheme [73, 74].

Packet loss rate: This rate indicates the rate of lost packets during their transmission period [75]. This metric reflects the congestion condition of the switch and the networks paths since the switches may be too busy and the interfaces may become overwhelmed and start to drop packet to process incoming packets causing them to be dropped [59, 76]. The packet loss of one path is the ratio of the number of lost packets calculated by subtracting the number of received packets from the number of transmitted ones and the number of transmitted packets.

Throughput: Throughput is the processing speed of the network's nodes and its performance. It is the quantity of data that has been processed and transmitted during a period of time from source to its destination (i.e. throughput = 1/responsetime). A successful load-balancing that allocates workload to nodes with the proper capacity maximizes throughput [77, 78].

Load Balancing in SDN Controllers

Multiple SDN controllers were proposed by the developers, each bearing their own implementation, their own platform and their own scalability degree. Some are proprietary and commercialized, others are opensource and collaborative. But all aim to construct a centralized global policy for the network instead of the per-hop flow management to achieve an optimal traffic management. In this section we discuss the input of some of the opensource controllers with regards to the load balancing issue. The Table 1 synthetize these controllers and whether they achieve the objectives of load balancing mentioned earlier whereas Table 2 classify them according to their architecture and programmation language.

POX: This controller is an SDN controller written in python [79]. The POX component Loadbalancer.py ensures load management and balances the incoming trafic between the network elements. The load balancing component redirects the incoming requests sent by the clients to the most appropriate https servers. The controller chooses the server based on three algorithms: Random selection, Round Robin algorithm and Weighted Round Robin.

Table 1 Load-balancing controllers performance

Controller	Scalability	Resilience	Rssource utilization	QoS
Beacon	✓	✓	×	×
Floodlight	✓	✓	✓	✓
Hyperflow	✓	✓	✓	✓
Onix	×	×	×	×
NOX	×	×	×	×
OpenDayLight	✓	✓	✓	✓
ONOS	×	×	×	×
POX	✓	×	✓	×

Table 2 Controllers classification

Controller	Architecture	Programmation language
Beacon	Physically centralized	Java
Floodlight	Physically centralized	Java
Hyperflow	Logically centralized, physically distributed	C++
Onix	Logically centralized, physically distributed	C, Python
OpenDayLight	Logically centralized, physically distributed	Java
ONOS	Logically centralized, physically distributed	Java
POX	Physically centralized	Python

Ryu: This software defined networking framework is written in python and uses versions 1.3 of openflow. It is an open source controller that aim to augment the agility of the network through using multiple components. The components have API's that make it easy to manage and adapt traffic handling to the network's objectives [80].

Floodlight: This controller is an opensource java-based Apache-licensed OpenFlow controller developed by David Erickson and a community of developers [81]. After enabling the statistics collection, the controller can perform load balancing using the Transmission Rate and the Receiving Rate of the data packet. Using the collected information, the best path is decided and the corresponding information (Port number, IP destination, IP source,...) are sent to the switches.

Opendaylight: This opensource controller is physically distributed and logically centralized and provides a rich set of basic and extended network services for network resource optimization [82]. OpenDayLight, through OVSDB integration, proposes the LOAD-BALANCER application.

The ODL load-balancer provides the ability to divide traffic reactively at the switch level to multiple end hosts. The controller can also determine how the flow will be forwarded proactively through hashing traffic by flow.

Hyperflow: This controller is logically centralized control plane and consists of many distributed controllers [35]. It's an evolution of the NOX controller that enables a multi-controller architecture which is logically centralized. It builds a global view of the entire network through the exchanged statistics and network data between controllers. Hyperflow operates smoothly and reactively under heavy load by taking traffic forwarding and synchronization decisions locally at each controller and achieves a low response time in comparison to NOX controllers.

Onix: This robust and scalable controller is logically centralized and physically distributed. It supports both the OpenFlow and OVSDB protocols [27]. It runs on a multiple physical servers simultaneously, and is allows control applications to consult and change the state of the network elements and to partition workload. Onix provides a good scalability through additional partitioning and aggregation mechanisms. Onix uses the network information base (NIB) data structure to store the state of the whole network which is distributed among Onix instances. Onix reduces the traffic load of a centralized controller by distributing network state among the switches. Onix also uses different mechanism to ensure failure recovery and reliability requirements .

Beacon: This opensource OpenFlow controller is based on Java [83–85]. Beacon supports event based and threaded operation. Beacon uses a method that statically fixes the number of switches assigned to a worker thread. It proved a better performance due to its capacity to process the network's flow using pipeline threads and shared queues. It also provides multiple applications each enabling a number of control functions. In [75], Beacon is enhanced and BeaQoS is proposed to efficiently perform queue load balancing. This solution manages and balances the queues of the SDN OpenFLow switches based on their flow rates and packet loss. Flow re-routing solves flows congestion in the network switches and improves their performance.

ONOS: Open network operating system (ONOS) is a physically distributed and logically centralized controller [86, 87]. ONOS also keeps track of the global network view to ensure network management and shares the network's state to all the servers in the cluster. ONOS runs on multiple servers, each one of them manages a group of switches. ONOS distributed architecture allows the network to avoid single point failure and it manages to pursue its activity even when one of its components fails by reassigning work to other remaining instances. In , the authors propose a dynamic load balancing application based on ONOS 1.3. The application runs on one of the c-nodes which are the servers that form the distributed controller cluster and collects the measurements results from all the others and based on the results configures an optimal weight parameter for the RR scheduling configured to the load balance.

Categorization of Load Balancing Algorithms in SDN

Centralized Load-Balancing

In SDN networks, the controller decides the forwarding path for each packet in the network through forwarding tables implemented by the data plane elements. Table 3 enlists some of the load-balancing methods in centralized software-defined networks. The authors of [88] present the fuzzy synthetic evaluation mechanism (FSEM). It's an SDN-based load balancing solution where the paths are dynamically calculated and adjusted according to the networks global view through flow-handling rules at the controller and using the fuzzy evaluation model. This mechanism starts by selecting the Top-K paths and then the best path is chosen. When the network has no traffic the TOP-K are chosen based on the Floyd algorithm and are the K shortest paths. When there is traffic, the Top-K paths are chosen by FSEM periodically to adapt to network changes.

FSEM is a multiple attribute fuzzy decision making algorithm that considers the length of the path measured by the number of hops, its load quantifies with byte and packet count of the critical switch and the link's traffic measured by its matching port's forwarding rate. The load balancing model proposed is composed of three modules: Data collection module, Path evaluation module constituted with Top-K and FSEM and Flow Table Installation module where the rules are installed by the controller based on the output of path evaluation. To evaluate this method the authors used POX controller. The experiments proved the effectiveness of this mechanism. The controller detects the faulty links instantly and selects a back-up path however the restoration of the traffic takes time and therefore some packets are lost.

The authors of [89] propose a load balancing algorithm for links based on ant colony optimization algorithm (LLBACO) and search rule. To select the next node, this algorithm takes into account link load, delay and packet-loss. The control plane is composed of four modules: monitor module, data collection, load-balancing and flow control module. LLBACO is deployed in the load-balancing module that computes the best end-to-end path for the packet-in messages. The algorithm sets a dynamic threshold to select the flow path and stores them according to the Ant Colony algorithm to finally compute the best path appended in the

Table 3 Load-balancing solutions in centralized controllers

Authors	Mechanism	Base network services	Results	Metrics
Li et al. [88]	Fuzzy synthetic evaluation mechanism (FSEM)	Select the best path out of the Top-*K* paths which are selected based either on FSEM when there is traffic or Floyd algorithm when there isn't	Efficient load balancing Instant failure detection Resilience and reliability	Latency Round trip time
Wang et al. [89]	Link load balancing algorithm based on ant colony optimization algorithm	The algorithm uses a dynamic threshold to select the flow path according to the ant colony search	Improves network overload Decreases transmission cost Stable packet-loss and delay	Latency Resource utilization Packet-loss rate
Agarwal et al. [90]	Fully polynomial time approximation schemes (FPTAS)	Dynamically manages traffic in a network where both SDN and OSPF routing is used based on network's topology information, available bandwidth and the amount of packet-ins	Better performance compared to OSPF in terms of: packet-loss rate Max/mean Latency	Latency Throughput Packet-loss rate
Carpio et al. [91]	DivFlow and ECMP	DivFlow is a multi-path routing scheme used for data-centers. It uses ECMP for short flows and random packet switching for long flows	Low FCT for short flows High throughput for long flows	Latency Throughput
Nkosiet et al. [92]	OpenDaylight multi-path load-balancing	Dynamic multi-flow load-balancing method based on Djikstra algorithm for path selection and traffic balancing	Decreased overall network delay Increased data transfer	Latency Throughput
Talikder et al. [93]	DTLB: dual threshold load balancing scheme	The algorithm uses two thresholds to which it compares the load condition to decide process migration The target physical machine is determined by the available memory of the server response time and measured load	Better response time Improved throughput Less memory usage	Response time Energy consumption Resource utilization

best PathList. Compared to other algorithms, LLBACO improves network overhead, has lower transmission cost and a stable packet-loss and delay.

In [90], the authors develop an SDN solution to dynamically manage traffic in a network using both SDN controllers and traditional hop-by-hop routing mechanisms. The system used is composed of forwarding elements and the controllers which peer with the network nodes to exchange topology information, available bandwidth and utilization of each link. The used standard routing protocol is OSPF. The experiments showed a significantly better performance of SDN Routing compared to the traditional routing protocols in terms of maximum and mean number of packets lost over all the links and per link and the maximum and mean delay.

The paper [91] presents DiffFlow which is a solution for data center used for mixed short and long flows. This solution combines the use of ECMP for short flows and the use of the load balancing method random packet switching (RPS) for long flows. For packets categorization, OpenFLow switches use packet sampling method periodically. The metrics used to measure the performance of this model are throughput and the flow completion time (FCT) which refers to the period between the instant the the first packet of a flow leaves a source server and the instant the last packet of the same flow arrives to its destination. This method's goal is to ensure a trade-off to guarantee a low FCT for short flows while providing an enhanced throughput for long flows.

In [92], the authors compare the performance of Opendaylight's load balancer as a centralized controller using both random policies and round robin algorithm, to a dynamic load balancer that uses a multiflow load balancing method based on Djikstra algorithm for path selection and traffic balancing. The controller extracts the transmission rate statistics for ports to evaluate the load on each port. The best shortest path is chosen based on lowest flow cost which is the sum of the number of transmitted and received packets at a given time. The load management algorithm is performed until equal path cost for all paths is achieved which indicates that they have the same load. The results showed that the maximum average ping after load balancing is 50% lower compared to its value before load balancing that lowers network's delay. Data transfer however increases significantly only when the controller has more alternative path options.

In [93] the authors present DTLB a load balancing scheme with two thresholds designed for SDN networks. The load balancing module is implemented in the controller to achieve load balancing by process migration in a cloud environment. It is programmed for centralized SDN networks and can be used in distributed systems. The used algorithm is executed in four steps: Process selection, physical machine selection, process assignment to physical machine and process migration. The current load of a physical machine is measured by the combination of response time and energy consumption, a high value indicates that the PM load is high. The scheme compares the collected system information to the algorithm threshold min and max to determine the load condition of the PMs. If the measure load is inferior to the minimum threshold it is underloaded, if it's superior to the max it is overloaded. In the first case the processes are migrated and the PM is put in sleeping state. In the second, the process that holds the higher process ID is the one selected to be migrated. The destination PM is selected based on the available memory on the server and the response time and measured load. The selected processes migrate to the selected physical machine by using the matching technique corresponding to the process-machine assignment. This method was evaluated based on throughput, energy consumption and memory usage. The results indicated a better response time and throughput compared to traditional models and four times less amount of memory usage.

Distributed Load Balancing

In [39] , the authors present ElastiCon , an elastic distributed SDN controller. This design works to shift dynamically the workload which allows the controllers to interfere at a prespecified load threshold. According to the load evolution and the maximum capacity of the existing controllers, the algorithm dynamically uses an expand or shrink resources pool. At high load, switches migrate from the controller carrying a heavy load to a less loaded controller. The packet processing rules are installed in the switches reactively in this paper. The load adaptation needs a periodic load balance of the controllers and it consists of estimating load, adaptation decision computation and finally migration action. To test the elastic distributed controller architecture, the authors developed a new openFlow based network emulator to run the OpenVSwitch instances on different hosts, it is an enhanced Mininet testbed. Since the subject of evaluation is the the control plane traffic load, the OpenVSwitches were modified to inject Packet-In messages to the controller without disabling the data plane. The results shows that adding controller nodes increases throughput. The response time increases when the generation rate of Packet-In messages increases indicating that the processor is in the bottle-neck but after migrating the switches until balancing the load between controllers it improves.

In [94], the author propose to enhance random neural network (RNN) with reinforcement learning to path selection based on the path congestion metric collected from network statistics in order to manage load distribution in data-centers. RNN model was introduced by Gelenbe [95], it is an interconnected network of neurones that exchange spiking signals. RNN provides adaptive and QoS driven routing to network packets [96]. This load balancing method is based on real time measurements, the cognitive packet network (CPN) protocol collects and measures the congestion level of every link with discounting rate estimator (DRE). The path congestion metric is the maximum of the link load metric, the port haven paths with the least congestion is selected. In [97], the authors propose a Smart Service Manager that uses RNN and reinforcement learning. The routing and server allocations decisions use measurement data based on machine learning. The objective of the solution is to choose the best node where the user services should be performed. This solution is able to provide a minimized overall average response time, improves quality of service, energy consumption and security.

In [98], the authors use multiple controllers each one of them is responsible to load balance a specific kind of application (Browsers, mails, ...) using a dedicated load balancing algorithm suited to each application. In [48], the authors present DALB, which is an adaptive distributed mechanism that manages traffic policies. In this algorithm each controller is responsible for a domain in the network. Each controller collects its load, other controllers load and adjusts the load collection threshold using an adaptive controller threshold algorithm which reduces the overhead of exchanging messages. Policies, elections and switch migrations are also made locally to reduce decision latency caused by network transmission and thus avoid single point failures. The used controllers are floodlight and can connect to multiple controllers using openflow considering that only one controller will be the master and the others will act as slaves. They are coordinating using Zookeper. The authors use the average incoming packets arrival rate to represent the load of the controller, and to choose the switch connected to the overloaded controller for migration. The selection of the migrated switch takes into consideration the number of flow table entries, the average message arrival rate and the RTT.

The authors of [99] propose a load informing strategy based load-balancing mechanism. The used network is composed of multiple distributed controllers using floodlight. The controllers periodically share their load informations and store the load informations shared by the other controllers, this is the load informing strategy. The load balancing module is composed of four components: Load measurement that determines the controller exceeding the predefined threshold, load informing , balance decision and switch migration components that select which switch needs to migrate and the target unloaded controller. To avoid migration conflict and target controller overload, when a switch migration is in process the target controller accepts a unique switch migration request. The experimental results showed a higher throughput compared to the static mapping model, a better completion time and a reduced load-balancing time.

Although switch migration based load balancing solves imbalances in a multi-controller network it can also increase network overhead and impact its performance. In [63], the authors propose a load-balancing scheme using algorithms that solve load changes and increase load-balancing time efficiency. The study is based on logically centralized control plane where the master controller and the other local controllers communicate and exchange decisions. The total load of controller is based on the arrival rate of the arriving packets. The switches load is represented by the sending rate of these packets. The balance decision composes of choosing switches selection and the target controller. The added-value of this research is that the switches selection algorithm takes into account the load of the heavily-loaded controllers and switches aiming the reduction of the time needed to achieve balance. The target controller is aimed to have a load near network average to avoid oscillation among controllers. To evaluate the proposed switches group mechanism, the authors compare it to the static controller-switch mapping and Elasticon. The results showed a better completion time and resource utilization when using the switches group method.

The SDN multiple controller load-balancing strategy based on response time (SSMCLBRT) [100] is a strategy based on response time designed for multiple controllers in a distributed SDN control plane. The response time threshold is continuously adjusted and multiple overloaded controllers are addressed simultaneously. In this algorithm, the response time of a single PACKET-IN event is the difference between the time arrival of PACKET IN message and the time of the PACKET-OUT or FLOW MOD reply that the controller sends. The workload of a switch in a determined period is the number of requests recorded in it. When the controller's load increases the response time follows, this calls for threshold calculation based on this variation feature. When the controller exceeds this threshold its needs to be processed rapidly. This algorithm compares all controllers response time to allow an efficient overloaded controllers identification and multiple load balance shiftings in a single detection. This scheme achieves a better overall response time, the load balancing operations starts earlier and it chooses the worst switch that impact the overloaded controller for migration to low-loaded controllers.

Flows loss or incorrect processing might occur during switches migration. To overcome these limitations, the authors in [101] propose SHLB a load balancing mechanism that inserts a plane between the control and the data plane to decrease the control plane over-utilization. Upon receiving the incoming packets from the data plane, the middle-plane forwards them to the controllers in the control-plane, then the results are sent back to the data-plane. The controllers in control-plane can be categorized as global or local. The global controller manages the load balancing for the control plane whereas the local ones send their load status to the global controller periodically and implement its rules and according to their state active or sleeping implements the requests from the middle-plane.

The global controller has two threshold, maximum and minimum to evaluate the load of the local controller and then run the load-balancing algorithm. A classifier is implemented in the added middle plane to classify the flow. To solve overload in middle plane each node is composed of three module: the first is collector which collects periodically the CPU utilization then sends it to the second module which is Judger that judges if it has abnormal load and in case it is informs the migrator which is the third module. A request is then sent to the master controller containing the IDs of the switches connected to the overloaded controller. This method which is divided into two main processes control's plane's load balancing and middle-plane's migration proved to reduce the network delay, improve stability and scalability of the network.

The work in [102] proposes a hierarchical load balancing mechanism for SDN environments composed of multiple controllers. The used control plane is implemented with a meta-control plane, which is composed of the global agent and the resource scheduler, that processes the resources usage of the local control plane to optimize their performance. The local controllers can be active or inactive according to the switches it manages. The active local controller is evaluated by the global agent to determine its load state, in case of load increase engender the resource scheduler activation to reschedule the local network elements, the local agent of the local controller receives the allocating request which is eventually implemented by the switch handler. The optimal controller scheduling is a linear programming problem, its objective function is to minimize the controller management cost and the SDN switch reassignment cost. The results showed a reduced congestion in the control plane and a bandwidth utilization increase.

In [103], the authors present a dynamic load balancing multi-controller deployment scheme that takes controllers capacity and traffic propagation delay as main impacting factors. A control-domain adjustment algorithm and an improved affinity propagation algorithm are proposed aiming to resolve the dynamic controller deployment issue and improve the clustering effect for a better network planning. The paper uses Breadth First Search algorithm to reassign switches in different sub-domains in order to ensure the load management of controllers and the traffic requests are transformed into a queuing model. The study aims to obtain a reasonable number of controllers and to optimize the switch-controller mapping relationships to minimize the overall communication costs and achieve a low latency between switches and controllers. This scheme, compared with affinity propagation and genetic algorithms, has a better load-balancing rate. All the mentioned distributed load-balancing methods are synthetized in Table 4.

Load Balancing in 5G Networks

5G networks are now being deployed around the world. The architecture of this new eco-system has major requirements and creates great challenges such as the management of the unprecedented growth of traffic, resource utilization optimization, quality of service and experience. 5G networks are especially sensitive to delay and latency since it supports real-time applications with tremendous data volumes [104, 105] which emphasises the importance of load-balancing as a user-centric networking technique.

Load-balancing is of essence in 5G networks [106] since they are designed in order to achieve a densified heterogeneous network architecture, where multiple RAN technologies are involved. This makes the use of load balancing for resource utilization optimization and to go along with the unprecedented growth of traffic [107]. A basic load-balancing algorithm is presented in [108]. To augment the 5G HetNets [109] with intelligence and control, load balancing algorithms for the SND-based heterogeneous network is proposed in [110, 111].

In [77], the authors tackle the load-balancing between different technologies enforced with flow admission control. This method offloads the core network, ensures a better resources utilization and avoids cellular-network's overutilization. In this solution traffic capacity is determined by the number of requested physical resources. The user with the highest PRB usage is regarded as overloaded and the network equalizes the load through network triggered intersystem handover. The aim is to optimize resources, minimize response time and maximize throughput. In [112], an algorithm that improves load-balancing throughout selecting the best network based on quality-of-service is introduced.

The authors of [113] propose an approach to improve MME scalability as a control plane entity in SDN and NFV based architecture. To achieve this goal, the authors improve the load balancing in a virtual and distributed MME architecture using both Round Robin (RR) and Weighted Round Robin (WRR). The results of the emulation showed that WRR had better resource utilization rate and lower attachment delay.

Access and mobility function AMF of the 5G core network is responsible for handling connection and mobility management. AMF receives its information from the users equipments. With the increase of users equipments (UEs), traffic over data and control plane of the network also increases which requires load-balancing. In [114, 115], linear quadratic regulator (LQR) controller is used. Periodically, the LQR will calculate the capacity of the AMF and broadcast it to the eNode-B to ensure even distribution of the incoming load among the AMF instances. In [116], the authors propose a utility-based load balancing

Table 4 Load-balancing solutions in distributed controllers

Authors	Mechanism	Base network services	Results	Metrics
Dixit et al. [39]	Elastic distributed controller architecture: elasticon	The system dynamically shrinks or extends the resource pool and reactively installs packet processing rules in the switches according to traffic conditions	Achieves load-balancing with minimal impact on response time Improves performance by increasing throughput	Response time Throughput
Frohlich et al. [94, 95, 96]	Random neural network (RNN) with reinforcement learning	Based on the path congestion metric, data center load is managed. The enhanced random neural network (RNN) provides adaptive and QoS driven routing	Resilience QoS oriented Reduced response time Security	Response time
Gelenbe et al. [97]	Self-aware computer systems and networks	A survey on the use of Artificial intelligence and Cognitive packet network to manage network's changes to achieve QoS objectives along with security, and energy	Resilience Reduced energy consumption Security	Latency Packet-loss rate
Zhou et al. [48]	Dynamic and adaptive algorithm for controller load balancing	Each controller is responsible for a domain in the network, migration decisions are made locally and the controllers coordinate using Zookeeper	Resilience Reduced overhead of exchanging messages by using an adaptive threshold algorithm Decreased throughput	Latency Overhead Forwarding entries
Yu et al. [99]	Load-informing based floodlight controller	Migration decision based on load informing strategy to avoid conflicts and target controller overload	Higher throughput compared to the static mode l Better completion timeReduced load-balancing time Increases network overhead	Throughput Completion time Overhead
Zhou et al. [63]	Switches group based on load balancing algorithm for multiple controllers architecture	The objective of this method is to avoid load oscillation in the network. The load of controllers is measured by the arrival rate of the arriving packets and by the sending rate of the switches	Better resource utilization Better completion time	Resource utilization Completion time
Cui et al. [100]	Response time based load balancing strategy for multiple controller SDN network (SMCLBRT)	The response time is continuously adjusted, controllers are simultaneously treated. A switch migration algorithm is activated when the load threshold is exceeded	Better response time Even load distribution Less time to deal with multiple overloaded controllers simultaneously	Response time Completion time
Xu et al. [101]	Scalable and hierarchical load balancing model: SHLB	A plane is added between data and control plane where a classifier is implemented to classify the flow to relieve the overload in the control-plane. The results showed that this mechanism has a significant improvement in network latency and stability	Reduced network delay Improved stability and scalability of the network	Latency Resource utilisation ratio Response time

Table 4 (continued)

Authors	Mechanism	Base network services	Results	Metrics
Ma et al. [102]	Meta-control based management mechanism	This mechanism ensures the load balancing of the local control plane to optimize network's data plane performance and avoids the congestion of the centralized control	Processing performance Greater system throughput Increased utilization of the bandwidth	Throughput Bandwidth utilization ratio
Li et al. [103]	Control-domain adjustment algorithm based on breadth first search (BFS)	CDAA based on BFS that performs the switch mapping changes and achieves an optimized load-balancing	CDAA reduces response time Better load-balancing rate	Latency Response time

algorithm, running on the load balancing controller (LBC), which can take both the RAN status and user needs into consideration. In fact, the fundamental concept is to get the status information of heterogeneous wireless access networks from the SDN controller southbound interfaces and accordingly install the appropriate load balance rules on the load balancer in the SDN controller northbound interfaces. The proposed algorithm is based on utility theory where the combination of user experience and network load with the maximum utility value is the one chosen. The experimental results showed that the load standard aberration declines and the RAN load becomes more balanced with time.

Conclusion

In this paper, we have presented a detailed analysis of the different algorithms and metrics used, in the literature, to ensure load balancing in both centralized and distributed SDN Architectures. Through the various studies, we can conclude that load balancing improves load distribution between the different SDN controllers. As a result, end users will have a better quality of service and of experience by reducing latency and response times and increasing throughput. Furthermore, software-defined networks based load balancing algorithms allow a global view of the network, so these methods generally improve network performance compared to traditional load balancing mechanisms. Nowadays, the SDN based load balancing solutions are used in different networks such as LTE, Cloud/fog, radio access network, and 5G networks. But in spite of it all, SDN paradigm still faces several challenges. The scalability of the controllers, their number and their location affects the network performance. The security of the controller is also an important issue, since there is a high security risk for it to be a single point failure in case of a successful attack. More researchers should be conducted to find the more appropriate locations for the controllers in the network, to develop more secured controllers and to secure the exchanges between them and towards other network components.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. McKeown N, Anderson T, Shenker S, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Turner J. Openflow: enabling innovation in campus networks. In: ACM SIGCOMM computer communication review. 2018.
2. Rowshanrad S, Abdi V, Namvarasl S, Hajizadeh M, Keshtgary M. A survey on SDN, the future of networking. J Adv Comput Sci Technol. 2014.
3. Mirchev A. Survey of concepts for QoS improvements via SDN. In: Future internet (FI) and innovative internet technologies and mobile communications (IITM), vol 33. 2015. p. 1.
4. Software defined networking: a new paradigm for virtual, dynamic, flexible networking. In: Hopewell junction, NY. White Paper. 2012
5. Brar A, Thapar V, Kishor K. A survey of load balancing algorithms in cloud computing. Int J Comput Sci Trends Technol (IJCST). 2014.
6. Hanamakkanavar AS, Handur VS. Load balancing in distributed systems: a survey. Int J Emerg Technol Comput Sci Electron (IJETCSE). 2015.
7. Vashistha J, Jayswal AK. Comparative study of load balancing algorithms. IOSR J Eng. 2013;3:45–50.
8. Sidhu AK, Kinger S, Sahib F. Analysis of load balancing techniques in cloud computing. Int J Comput Technol 2013;4.
9. MacVittie D. Intro to load balancing for developers the algorithms. DevCentral. [Online] 31 March 2009. https://devcentral.f5.com/articles/intro-to-load-balancing-for-developers-ndash-the-algorithms.
10. Bala K, Vashist S, Singh R, Singh G. A review of the load balancing techniques at cloud server. Int J Adv Comput Sci Commun Eng. 2014;2:2347–6788.
11. Rosen E, Callon R, Viswanathan A. Multiprotocol label switching architecture. In: IETF RFC 3031. 2001.
12. Thaler D, Hopps C. Multipath issues in unicast and multicast next-hop selection. In: IETF RFC 2991. 2000.
13. Toguyeni A, Korbaa O. DiffServ aware MPLS traffic engineering for ISP networks: state of the art and new trends. J Telecommun Inf Technol. 2009.
14. Cao Z, Wang Z, Zegura E. Performance of hashing-based schemes for internet load balancing. In: Proceedings of the 19th annual joint conference on computer and communications societies. 2000.
15. Long K, Whang Z, Cheng S. Load balancing algorithms in MPLS traffic engineering. In: Proceeding of IEEE international conference on high performance switching and routing, Dallas, 2011.
16. Haleplidis E, Pentikousis K, Denazis S, Hadi Salim J, Meyer D, Koufopavlou O. Software-defined networking (SDN): layers and architecture. In: Terminology. 2015.
17. Cox J, Chung M, Joaquin F, Donovan S, Ivey J, Clark RJ, George R, Henry OL. Advancing software-defined networks: a survey. In: IEEE Access. 2017.
18. Lin W, Zhang L. The load balancing research of SDN based on ant colony algorithm with job classification. In: Proc. 2nd workshop adv. res. technol. ind. appl. 2016.
19. Nunes B, Mendonca M, Nguyen XN, Obraczka K, Turletti T. A survey of software-defined networking: past, present, and future of programmable networks. In: IEEE communications surveys & tutorials. 2014.
20. Nygren A et al. OpenFlow switch specification 1.5.0. 2014.
21. Yu M, Rexford J, Freedman MJ, Wang J. Scalable flow-based networking with DIFANE. In: ACM SIGCOMM computer communication review—SIGCOMM '10, vol 40, issue 4. 2010.
22. Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. NOX: towards an operating system for networks. In: ACM SIGCOMM computer communication review, vol 38, no 3. 2008.
23. Al Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera. Dynamic flow scheduling for data center networks. In: Proc. 7th USENIX conf. netw. syst. design implement. 2010. p. 19.
24. Li Y, Pan D. OpenFlow based load balancing for fat-tree networks with multipath support. In: IEEE international conference on communication (ICC). 2013.
25. Berde P, Gerola M, Hart J, Higuchi Y, Kobayashi M, Koide T, Lantz B, O'Connor B, Radoslavov P, Snow W, et al. Onos: towards an open, distributed SDN OS. In: Proceedings of ACM on hot topics in software defined networking (HotSDN), Chicago. 2014.
26. Canini M, Kuznetsov P, Levin D, Schmid S. A distributed and robust sdn control plane for transactional network updates. In: Proceedings of IEEE INFOCOM HongKong. 2015.
27. Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, Ramanathan R, Iwata Y, Inoue H, Hama T et al. Onix: a distributed control platform for large-scale production networks. In: Proceedings of USENIX operating systems design and implementation (OSDI). 2010.
28. Shalimov A, Zuikov D, Zimarina D, Pashkov V, Smeliansky R. Advanced study of sdn openflow controllers. In: Proceedings of the 9th Central; Eastern European software engineering conference in Russia. 2013.
29. Karakus M, Durresi A. A survey: control plane scalability issues and approaches in software defined networking (sdn). Comput Netw. 2017.
30. Wang R, Butnariu D, Rexford J. Openflow-based server load balancing gone wild. In: Proc. USENIX HotICE. 2011.
31. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. Openflow: enabling innovation in campus networks. In: ACMSIGCOMM. 2008.
32. Campbell A, Katzela I, Miki K, Vicente J. Open signaling for atm, internet and mobile networks (opensig'98). In: ACM SIGCOMM. 1999.
33. Moore J, Nettles S. Towards practical programmable packets. In: Proceeding of IEEE computer and communication societies (INFOCOM) Anchorage. 2001.
34. Yu M, Rexford J, Freedman MJ, Wang J. Scalable flow-based networking with DIFANE. SIGCOMM Comput Commun Rev. 2010;41(4):351–62.
35. Tootoonchian A, Ganjali Y. HyperFlow: a distributed control plane for OpenFlow. In: Proceedings of the 2010 internet network management conference on research on enterprise networking. USENIX Association; 2010. p. 3.
36. Cai Z. Maestro: achieving scalability and coordination in centralized network control plane. Ph.D. dissertation. Rice Univ., Houston. 2011
37. Voellmy A, Wang J. Scalable software-defined network controllers. In: ACM SIGCOMM 2012 conference on applications, technologies, architectures, and protocols for computer communication. 2012. p. 289–90.
38. Sezer S, et al. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Commun Mag. 2013;51(7):36–43.
39. Dixit A, Hao F, Mukherjee S, Lakshman TV, Kompella RR. ElastiCon: an elastic distributed SDN controller. In: ACM/IEEE symposium on architectures for networking and communications systems (ANCS), Marina del Rey. 2014.
40. Yeganeh S, Tootoonchian A, Ganjali Y. On scalability of software-defined networking. IEEE Commun Mag. 2013;51(2):136–41.

41. Borcoci E, Badea R, Obreja SG, Vochin M. On multi-controller placement optimization in software defined networking-based WANs. In: ICN 2015: the fourteenth international conference on networks.
42. Xiao P, Qu W, Qi H, Li Z, Xu Y. The SDN controller placement problem for WAN. In: IEEE/cic international conference on communications in China (ICCC). IEEE. 2014. p. 220–24.
43. Pfeiffenberger T, Du JL, Arruda PB, Anzaloni A. Reliable and flexible communications for power systems: fault-tolerant multicast with SDN/OpenFlow. In: 7th International Conference on new technologies, mobility and security (NTMS). IEEE. 2015. p. 1–6.
44. Muller L, Oliveira R, Luizelli M, Gaspary L, Barcellos M. Survivor: an enhanced controller placement strategy for improving SDN survivability. In: IEEE global comm. conference (GLOBECOM). 2014.
45. Zhang X, Phillips C. Network operator independent resilient overlay for mission critical applications (ROMCA). In: Fourth international conference on communications and networking in China COM. IEEE. 2009. p. 1–5.
46. Dan M, Lei L, Yoo SJB. Optical FlowBroker: load-balancing in software-defined multi-domain optical networks. In: Optical fiber communication conference. 2014. p. W2A.44.
47. Hai NT, Kim DS. efficient load balancing for multi-controller in SDN-based mission-critical networks. In: IEEE international conference on industrial informatics. 2016.
48. Zhou Y, Hu M, Xiao L, Ruan Li, Duan W, Li D, Liu R, Zhu M. A load balancing strategy for SDN controller based on distributed decision. In: IEEE 13th international conference on trust, security and privacy in computing and communications. 2014.
49. Hu Y, Wang W, Gong X et al. BalanceFlow: controller load balancing for OpenFlow networks. In: IEEE international conference on cloud computing and intelligence systems. 2012. p. 780–85.
50. Adami D, Giordano S, Pagano M, Santinelli N. Class-based traffic recovery with load balancing in software-defined networks. In: Globecom 2014 workshop—the 6th IEEE international workshop on management of emerging networks and services.
51. Hamed MI, ElHalawany BM, Fouda MM, Eldien AST. A novel approach for resource utilization and management in SDN. In: 13th international computer engineering conference (ICENCO2017).
52. Tuncer D, Charalambides M, Clayman S, Pavlou G. Adaptive resource management and control in software defined networks. In: IEEE transactions on network and service management, vol 12, no 1. 2015
53. Zaouch A, Benabbou F. Load balancing for improved quality of service in the cloud. Int J Adv Comput Sci Appl. 2015;. https://doi.org/10.14569/IJACSA.2015.060724.
54. James J, Verma B. Efficient VM load balancing algorithm for a cloud computing environment. Int J Comput Sci Eng. 2012.
55. Moharana SS, Ramesh RD, Powar D. Analysis of load balancers in cloud computing. Int J Comput Sci Eng (IJCSE). 2013;2(2):101–8.
56. Jogalekar P, Woodside M. Evaluating the scalability of distributed systems. IEEE Trans Parallel Distrib Syst. 2000;11(6):589–603. https://doi.org/10.1109/71.862209.
57. Hu J, Lin C, Li X, Huang J. Scalability of control planes for Software defined networks: modeling and evaluation. In: IEEE 22nd international symposium of quality of service (IWQoS), Hong Kong. 2014. p. 147–52. https://doi.org/10.1109/IWQoS.2014.6914314.
58. Bouzghiba S, Dahmouni H, Rachdi A. GJM.arcia, Towards an autonomic approach for software defined networks: an overview. In: Advances in ubiquitous networking 2. Springer. 2017. p. 149–61.
59. Chen-xiao C, Ya-bin X. Research on load balance method in SDN. Int J Grid Distrib Comput. 2016.
60. Sharma S, Singh S, Sharma M. Performance analysis of load balancing algorithms. In: World academy of science, engineering and technology. 2008. p. 38.
61. Hai NT, Kim DS. Efficient load balancing for multi-controller in SDN-based mission-critical networks. In: Proc. IEEE 14th Int. Conf. Ind.Inform. (INDIN). 2016. p. 420–25.
62. Song P, Liu Y, Liu T, Qian D. Flow stealer: lightweight load balancing by stealing flows in distributed SDN controllers. Sci China Inf Sci. 2017;60(3):032202.
63. Zhou Y, Wang Y, Yu J, Ba J, Zhang S. Load balancing for multiple controllers in SDN based on switches group. In: APNOM2017. IEEE. 2017
64. Liu J, Li J, Shou G, Hu Y, Guo Z, Dai W. SDN based load balancing mechanism for elephant flow in data center networks. In: International symposium on wireless personal multimedia communications (WPMC2014).
65. Hu Y, Wang W, Gong X, Que X, Cheng S. BALANCEFLOW: controller load balancing for OpenFLow netwoks. In: IEEE CCIS2012.
66. Lin YD, Wang CC, Lu YJ, Lai YC, Yang HC. Two-tier dynamic load balancing in SDN-enabled Wi-Fi networks. Wirel Netw. 2017;23:1–13.
67. Zhu R, Wang H, Gao Y, Yi S, Zhu F. Energy saving and load balancing for SDN based on multi-objective particle swarm optimization. In: Proc. Int. Conf. Algorithms Archit. Parallel Process. 2015. p. 176–89.
68. Liao W, Kuai S, Lu C. Dynamic load-balancing mechanism for software-defined networking. In: International conference on networking and network applications. 2016.
69. Dixit A, Hao F, Mukherjee S, Lakshman TV, Kompella R. Towards an elastic distributed SDN controller. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking—HotSDN '13.
70. Kaur H. Traffic based load balancing in software defined networking. Int J Comput Sci Eng (IJCSE)
71. Zhong H, Fang Y, Cui J. LBBSRT: an efficient SDN load balancing scheme based on server response time. Future Gener Comput Syst. 2017;68:183–90.
72. Vinayagamurthy D, Balasundaram J. Load balancing between controllers. University of Toronto. 2012.
73. Zhong H, Fang Y, Cui J. LBBSRT: an efficient SDN load balancing scheme based on server response time. Future Gener Comput Syst. 2015;80:409–16.
74. Wenbo C, Shang Z, Xinning T, Hui L. Dynamic server cluster load balancing in virtualization environment with OpenFlow. Int J Distrib Sens Netw. 2015.
75. Boero L, Cello M, Garibotto C, Marchese M, Mongelli M. yBeaQoS: load balancing and deadline management of queues in an OpenFlow SDN switch. Comput Netw. 2016;106:161–70.
76. Nam H, Kim KH, Kim JY, Schulzrinne H. Towards QoE-aware video streaming using SDN. In: 2014 IEEE global communications conference.
77. Namal S, Ahmad I, Gurtov A, Ylianttila M. SDN based inter-technology load balancing leveraged by flow admission control. In: 2013 IEEE SDN for future networks and services (SDN4FNS)
78. Daraghmi EY, Yuan S-M. A small world based overlay network for improving dynamic load-balancing. J Syst Softw. 2016;107:187–203.
79. POX. https://www.noxrepo.org/pox/about-pox/.
80. Ryu SDN Framework. 2017. https://osrg.github.io/ryu/.
81. FLoodlight project. https://www.projectfloodlight.org/floodlight/.

82. OpendayLight project. https://www.opendaylight.org/.
83. Bannour F, Souihi S, Mellouk A. Distributed SDN control: survey, taxonomy, and challenges. In: IEEE communications surveys and tutorials. 2018.
84. Beacon. https://openflow.stanford.edu/display/Beacon/Home.
85. Erickson D. The Beacon OpenFlow controller. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined net working, ser. HotSDN '13. ACM. 2013.
86. Berde P, Gerola M, Hart J, Higuchi Y, Kobayashi M, Koide T, Lantz B, O'Connor B, Radoslavov P, Snow W, Parulkar G. Onos: toward an open distributed sdn os. In: Proceedings of the third work-shop on hot topics in software defined networking, ser. HotSDN '14. ACM. 2014.
87. Muqaddas AS, Bianco A, Giaccone P, Maier G. Inter-controller traffic in onos clusters for sdn networks. In: 2016 IEEE international conference on communications (ICC). 2016. p. 1–6
88. Li J, Chang X, Ren Y, Zhang Z, Wang G. An effective path load balancing mechanism based on SDN. In: IEEE 13th international conference on trust, security and privacy in computing and communications, Beijing. 2014. p. 527–33.
89. Wang C, Zhang G, Xu H, Chen H. An ACO-based link load-balancing algorithm in SDN. In: 7th International conference on cloud computing and big data (CCBD), Macau. 2016. p. 214–18.
90. Agarwal S, Kodialam M, Lakshman TV. Traffic engineering in software defined networks. In: Proceedings IEEE INFOCOM. 2013.
91. Carpio F, Engelmann A, Jukan A. DiffFlow: differentiating short and long flows for load balancing in data center networks. In: Proc. IEEE GLOBECOM, vol. 10. 2016. p. 1–6.
92. Nkosi M, Lysko A, Ravhuanzwo L, Nandeni T, Engelberencht A. Classification of SDN distributed controller approaches: a brief overview. In: International conference on advances in computing and communication engineering (ICACCE). 2016.
93. Talukder A, Abedin SF, Munir MS, Hong CS. Dual threshold load balancing in SDN environment using process migration. In: International conference on information networking (ICOIN), Chiang Mai. 2018. p. 791–96.
94. Liu P. Using random neural network for load balancing in data centers. In: Proceedings on the international conference on internet computing (ICOMP). The steering committee of the world congress in computer science, computer engineering and applied computing (WorldComp). 2015. p. 3
95. Gelenbe E. Random neural networks with negative and positive signals and product form solution. Neural Comput. 1989;1(4):502–10.
96. Fröhlich P, Gelenbe E, Nowak MP. SmartSDN management of fog services. In: Proc. Global IoT summit (GIoTS), Dublin, IEEE Commun. Soc. 2020. p. 1–6.
97. Gelenbe E, Domańska J, Fröhlich P, Nowak M, Nowak S. Self-aware networks that optimize security, QoS, and energy. In: Proceedings of the IEEE. 2020. p. 1–18. https://doi.org/10.1109/JPROC.2020.2992559.
98. Koerner M, Kao O. Multiple service load-balancing with openflow. In: High performance switching and routing 13th international conference. 2012.
99. Yu J, Wang Y, Pei K, Zhang S, Li J. A load balancing mechanism for multiple SDN controllers based on load informing strategy. In: The 18th Asia-Pacific network operations and management symposium (APNOMS). 2016.
100. Cui J, Lu Q, Zhong H, Tian M, Liu L. A load-balancing mechanism for distributed SDN control plane using response time. IEEE Trans Netw Serv Manag. 2018;15(4):1197–206.
101. Xu Q, Li L, Liu J, Zhang J. A scalable and hierarchical load balancing model for control plane of SDN. In: 6th international conference on advanced cloud and big data (CBD), Lanzhou. 2018. p. 6–11.
102. Ma YW, Chen JL, Tsai YH et al. Load-balancing multiple controllers mechanism for software-defined networking. Wirel Pers Commun. 2017.
103. Li G, Wang X, Zhang Z. SDN-based load balancing scheme for multi-controller deployment. IEEE Access. 2019;7:39612–22.
104. GSMA. An introduction to network slicing. https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf.
105. Campolo C, Molinaro A, Iera A, Fontes R, Rothenberg CE. Towards 5G network slicing for the V2X ecosystem. In: 2018 4th IEEE conference on network softwarization and workshops (NetSoft). 2018. p. 400–405.
106. Gudipati A, Perry D, Li LE, Katti S. SoftRAN: software defined radio access network. In: ACM SIGCOMM 2nd workshop on hot topics in software defined networking (HotSDN). 2013.
107. Maksymyuk T, Dumych S, Brych M, Satria D, Jo M. An IoT based monitoring framework for software defined 5G mobile networks. In: Proceedings of the 11th international conference on ubiquitous information management and communication. 2017
108. Kwan R, Arnott R, Paterson R, Trivisonno R, Kubota M. On mobility load balancing for LTE systems. IEEE Veh Technol Conf. 2010;1(5):6–9.
109. Duan X, Akhtar AM, Wang X. Software-defined networking based resource management: data offloading with load balancing in 5 G HetNet. EURASIP J Wirel Commun Netw. 2015.
110. Xiaoyu D, Akhtar A, Wang X. Software-defined networking-based resource management: data offloading with load balancing in 5G HetNet. EURASIP J Wirel Commun Netw. 2015.
111. Rahimi H, Zibaeenejad A, Safavi A. A novel IoT architecture based on 5G-IoT and next generation technologies. In: IEEE 9th annual information technology, electronics and mobile communication conference (IEMCON). 2018. p. 81–8.
112. Haydar J, Ibrahim A, Pujolle G. A new access selection strategy in heterogeneous wireless networks based on traffic distribution. In: Wireless Days 1st IFIP. 2008.
113. Nguyen V, Brunstrom A, Grinnemo K, Taheri J. SDN/NFV-based mobile packet core network architectures: a survey. IEEE Commun Surv Tutor. 2017;19(3):1567–602.
114. Alawe I, Hadjadj-Aoul Y, Ksentini A, Bertin P, Darche D. On the scalability of 5g core network: the AMF case. In: IEEE CCNC. 2018
115. Alawe I, Hadjadj-Aoul Y, Ksentini A, Bertin P, Darche D, Cesar V, Davy D. Smart scaling of the 5G core network: an RNN-based approach. In: EEE global communications conference, GLOBECOM. 2018.
116. Tang W, Liao Q. An SDN-based approach for load balance in heterogeneous radio access networks. In: IEEE symposium on computer applications and communications, Weihai. 2014.