Telecommunication Systems (2022) 81:475–502
https://doi.org/10.1007/s11235-022-00947-6

Taxonomy of traffic engineering mechanisms in software-defined networks: a survey

Ramin Mohammadi[1] · Sedat Akleylek[1] · Ali Ghaffari[2] · Alireza Shirmarz[3]

Accepted: 29 July 2022 / Published online: 5 September 2022

Abstract
Nowadays, many applications need varying levels of Quality of Service (QoS). The network that provides the communication service connects the servers and clients. The network traffic which is routed through the network should be engineered. Traffic Engineering (TE) is a mechanism for transferring the packets considering the different QoS level requirements among applications. The optimal resource allocation is the primary strategy for TE so that the network can provide the QoS requirements for each application. The TE can improve network efficiency, performance, and user satisfaction. Software Defined Network (SDN) has been proposed as the novel network architecture that could make networks agile, manageable, and programmable using control and data plane separating compared to traditional network architecture. In this paper, we survey network traffic engineering in SDN. We investigate and cluster the articles published between 2017 and 2022 on traffic engineering in SDN. The state-of-the-art articles about the traffic engineering mechanisms in SDN have been examined and classified into four types: topology discovery, traffic measurement, traffic load balancing, QoS, and dependability. Finally, the cutting-edge issues and challenges are discussed for future research in SDN-based TE.

Keywords SDN · Traffic engineering · Traffic measurement · Traffic management · Load balancing · QoS

1 Introduction

There is a vast spectrum of research on the network topics such as cloud computing, Wireless Sensor Networks (WSNs) [1]2], Internet of Things (IoT) [3]4], big data applications [5], VoIP [6], and data centers services [7]. Many companies have invested tremendous capital in data centers and inter-datacenters networks [8]. The traditional network cannot provide different QoS levels for each application like video streaming, VoIP, web, and others; therefore, it needs another architecture [9]. There are many types of services, such as video conferencing [10], distance learning, online gaming, and e-commerce [11], that need real-time and enormous multimedia traffic (audio, video, and data). Network management and load balancing have become more complex in the traditional network architecture [12]. TE is a critical challenge in large networks and the Internet [13], 14]. It is a highly effective mechanism for improving the efficiency of data transmission networks; it operates by reducing end-to-end delay, congestion [15], energy consumption [16], 17 [17], packet loss, and enhances Quality of Experience (QoE). It dynamically analyzes, predicts, and adjusts network behaviour to achieve optimizations and improvements in data transmission. The essential and fundamental prerequisite of offering guaranteed QoS is accurate traffic prediction, monitoring, and measurement techniques [18]. The novel SDN is used for TE to be implemented in some pieces of the research, as mentioned in [19], 20, 21]. The merits and features of SDNs are as follows: (1) Control planes and data

✉ Ali Ghaffari
A.Ghaffari@iaut.ac.ir

Ramin Mohammadi
ram_moh1@yahoo.com

Sedat Akleylek
sedat.akleylek@bil.omu.edu.tr

Alireza Shirmarz
A.shirmarz@iau-tnb.ac.ir

1 Department of Computer Engineering, Ondokuz Mayis University, 55139 Samsun, Turkey

2 Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

3 Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran

planes are separate from one another. (2) Network is controlled and managed in a centralized and integrated manner to obtain a comprehensive view of the network status. (3) The network is programmable via software applications. (4) Forwarding decisions are used instead of destination-based and flow-based decisions.

1.1 Software Defined Network (SDN)

Software defined network is an architecture that separates the control plane from the data plane. This architecture consists of three layers: data, control, and application. SDN has three APIs named northbound, southbound, and east–west. Northbound API is an interface for connecting the network applications and control layer. Southbound API connects the data and control layer in SDN. East–west API is considered for the control layer scalability [22], 23, 24]. SDN architecture is illustrated in Fig. 1 briefly.

Each layer has a predefined task to make the network programmable and agile with collaboration. These responsibilities have been discussed based on the three main layers apart in the following.

1.1.1 Data-plane layer

The lowest layer in this architecture contains only Forwarding Elements (FE). Each FE is only responsible for forwarding because it is independent of network functionalities such as routing, switching, and firewalling. FE forwards the flows, a sequence of packets with a common source and destination, according to the flow table placed in FE. The flow table includes the flow rules; that is, the flow entity with the action that FE should do, such as forwarding to the controller, forwarding to the specific port, flooding, or discarding. The action is designated in the controller and exported to the FE [23], 25, 26].

1.1.2 Control plane layer

This layer plays an outstanding role in decision-making so that it can be reckoned as the brain of the network [27]. The decisions are made in this layer and executed using southbound API like Openflow. These decisions are the flow rules, including flow features and the action. These rules are constituted in the control layer and exported to each FE. These rules are the source of the flow table in each FE. The flow table is the basis of the FE's forwarding. East–west API has been designed to scale up the control plane from the number of controllers and their placement. [24]28.

1.1.3 Application layer

Many network applications are required; hence, this layer facilitates the architecture to develop network applications. This layer communicates with the control layer using northbound API because it should execute many commands utilizing the control layer [22], 23, 26].

1.2 Network traffic engineering

The programmability and agility in SDN have motivated many traditional network equipment vendors such as NEC, Juniper, Cisco, and HP to produce SDN-based equipment that can support the southbound API Openflow. Many software companies like Google and Microsoft have implemented their SDN-based data centers [23]. TE is one of the significant network functionalities developed in SDN. They can use this function to provide the optimal path for each request that each FE has requested. The controller can find and allocate the proper path to each flow request [23] 29, 30]. The main problem debated about TE is the resource allocation strategy that can provide the flow requirements. This strategy makes a facility for each network to transfer any flow while supporting the required QoS level of each application.

1.3 Paper structure

This paper addresses traffic engineering in a software-defined network and strives to classify the solutions proposed in academic research papers. Traffic engineering mechanisms and technologies are stated briefly, and their advantages and disadvantages are discussed in this article. Also, the contribution of traffic engineering in SDN and its role in eliminating the shortcomings of traditional networks are discussed. This article provides a novel classification for investigating the impact of SDN architecture on traffic engineering. On the other hand, the mutual effect and interaction of SDN and traffic engineering are explained in terms of topology discovery, traffic load balancing, measurement, QoS provisioning, and dependability. This review addresses traffic engineering mechanisms related to each classification. The reviewed papers will be compared with each other. This paper aims to state the merits and demerits of each approach. Finally, Challenges and open research problems regarding traffic engineering in SDN are highlighted and reviewed.

The significant contributions of the present overview paper include the following:

1. The role and significance of traffic engineering in SDNs are discussed.
2. A comprehensive and thematic categorization of traffic engineering is provided where different issues such as topology discovery, load balancing, QoS provisioning,

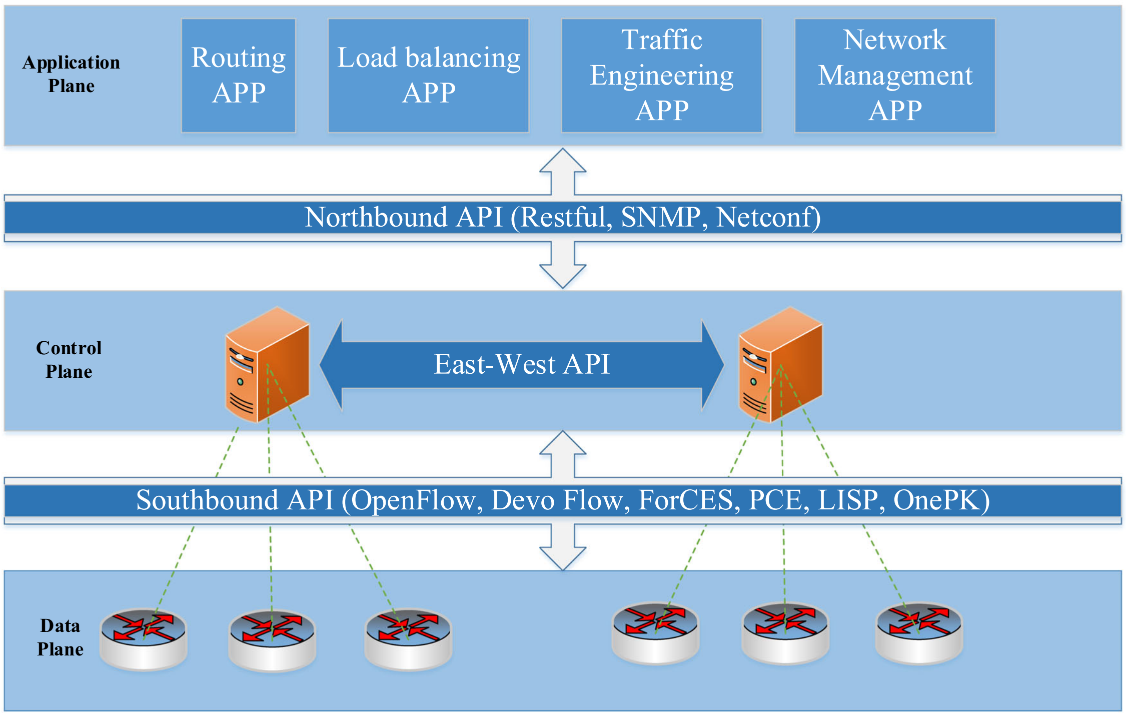

Fig. 1 Software Defined Network Architecture [23]

traffic measurement, traffic analyses, and dependability are considered.

3. Potential research gaps, challenges, and directions for further traffic engineering in SDN are provided.

The rest of the paper is organized as follows: Sect. 2 pays to similar previous survey articles. In Sect. 3, we address SDN's traffic engineering mechanisms taxonomy. The future research and issues of TE in SDN are discussed in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Related Work

Some papers have addressed a survey of TE in SDN. We try to examine the more important of them. A. Ghaffari et al. have worked on congestion control and surveyed the solutions to avoid congestion [1]. M. Abbassi et al. have worked on the review of TE in SDN [31]. The authors examined the routing that can engineer the traffic dynamically. P. Siripongwutikorn et al. have surveyed the balanced load routing to engineer traffic [32]. M. Karakus et al. examined the solutions proposed for QoS in SDN [9]. Z. Abdullah et al. have reviewed the papers about segment routing in SDN [33]. They have worked on segment routing, in which TE is one of the most significant applications. Traffic engineering is used among inter segments and ntra segments discussed. I. Bouleanu et al. have worked on network planning and traffic engineering on deployable networks [34]. M. Priyadarsini et al. have addressed software defined networking architecture, traffic management, security and placement in a survey [35]. They have focused on load balancing and energy-efficient routing, SDN control implementation and deployment architecture, controller security, and optimal controller placement that affects traffic management. These papers have worked on the issues and solutions proposed for traffic engineering and SDN. This paper strives to classify network traffic engineering in SDN into five classes: topology discovery, traffic measurement, load balancing, QoS, and dependability. This classification can cover many goals that many researchers have achieved using traffic engineering in software defined networks in their published papers.

3 Network traffic engineering mechanisms taxonomy in SDN

Network architecture can be divided into traditional & SDN, so this paper addresses the SDN. In SDN, network functionalities are based on flow, a sequence of packets with the same source and destination. Traffic engineering is a technique to lead the flows smartly. TE is done based on the network resource to cover non-functional requirements like performance, QoS, dependability, and so on. In SDN, two types of traffic can be considered for traffic engineering: the data

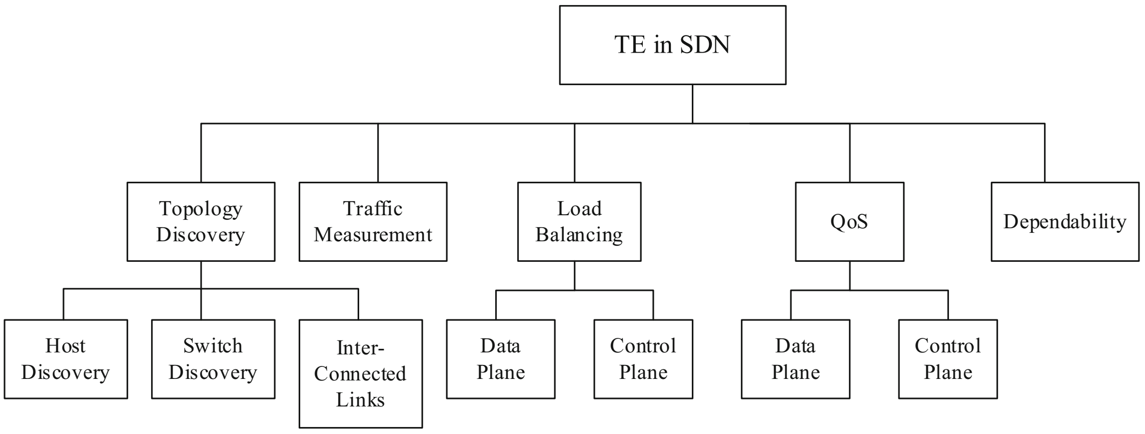

Fig. 2 Taxonomy of TE mechanisms in SDN

plane and distributed control plane. The controller plays a critical role as the network brain in SDN; hence, it can do traffic engineering. In addition, the distributed control layer has been proposed to make the SDN scalable; therefore, the network traffics among the controllers needs to be managed to cover the control traffic requirements.

Network TE in SDN will be surveyed based on the technique types in the continual subsection. The primary purpose of traffic management is to investigate how to manage network traffic based on network status information to satisfy users' needs, such as QoS in network applications. For fulfilling this purpose, in addition to discovering network topology, network status information should be obtained through traffic measurement technologies; then, the operation of traffic load balancing should be carried out optimally and efficiently. Hence, different traffic engineering mechanisms specify network topology and collect traffic information from network equipment. In this way, the QoS of applications is met. In this section, different traffic engineering mechanisms in SDN are categorized (topology discovery mechanisms, traffic measurement mechanisms, load balancing mechanisms, QoS provisioning mechanisms, and dependability), investigated and compared with one another. We explain these categories and related studies in corresponding sections. Figure 2 illustrates this classification.

The TE mechanism in SDN will be examined and expressed in detail in the following section.

3.1 Topology discovery

The network topology should be accurately determined for better management and precise network traffic analysis for operations such as routing, diagnosing, mobility tracking, resource management, monitoring, and load balancing [36]. In SDN, the controller needs to discover the SDN network topology [37][38]. Moreover, network topology in SDN encounters several problems and issues [39][40]. SDN's

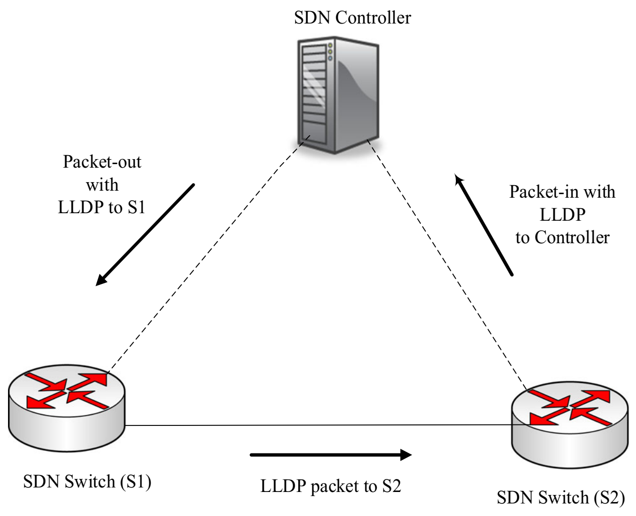

Fig. 3 LLDP in SDN

control and application plane should discover network topology to fulfill their purposes. A centeralized controller is embedded in SDN architecture to extract the entire network topology. This global view in the SDN controller can improve the routing and resource allocation mechanism compared to traditional networks. Three modules exist in SDN which should be discovered, including host, switch, and links.

3.1.1 Host discovery

For optimal network management, network topology should be determined, including the number and types of devices and their connections. Topology management (discovery and update) is SDN's unique and essential term compared to traditional networks. With appropriate and accurate topology discovery in each network, we can control and improve the network operation such as fault detection and recovery, congestion detection and mitigation, accounting, and security. In

SDN, the controller is responsible for discovering the number and the type of hosts using the host tracking function. Consequently, by finding hosts, it gets information about the precise location of the hosts in the network, which obtains information about network topology. This operation results in the accurate monitoring of network traffic, data routing, and the specification of the source of data packets [41]. The controller keeps a host profile table for each host, and if it leaves the network, it removes its related host controller table. The controller constitutes the flow table using received Packet-In messages [42]. Port and switch ID numbers are exchanged if the host migrates from one network switch to another. Upon receiving a Packet-In message from a new position, the controller updates the table related to that host.

3.1.2 Switch discovery

When OpenFlow switches receive a new packet, they communicate with the controller by transmitting Packet-In. In turn, the controller maintains communications with network switches by sending Packet-out messages. Hence, the controller specifies the positions of network switches in the network via the hand-shaking method. As the added switch to the network is determined, the controller registers the information related to that switch, such as MAC address and number of ports. Researchers in [71] and [72] made changes and modifications to the OFD mechanism to reduce controller overhead to discover network topology. The proposed model reduced the number of packet-out messages to improve efficiency. By implementing the optimized method on the POX controller and Mininet emulator [75], the overhead of the controller was reduced significantly.

In [43], the authors proposed a distributed algorithm based on a straightforward, simple agent-based method to optimize the efficiency of the topology discovery process. This algorithm was applied to design a novel topology discovery protocol called software-defined network-topology discovery protocol (SD-TDP). This protocol was implemented in each OpenFlow switch using a software agent. As a result, this method puts forth a distributed solution in which nodes that support the network protocol execute the topology discovery process.

3.1.3 Inter connected links

Switches maintain communications with one another through communication links. The identification of communication links between switches is essential for discovering topology. OFDP protocol is used for identifying communication links between switches. OFDP protocol uses LLDP (link layer discovery protocol) to distribute information related to the node's neighbours in the network [44]. SDN controller transmits LLDP packets as Packet-out messages to all the network switches. As switches receive the LLDP packet from the controller, they are sent to all the network switches that directly communicate with it. When switches receive LLDP packets from other network switches, they transmit Packet-In to the controller. Upon receiving the Packet-In from the network switches by the controller, the controller analyzes the respective packet and determines the directly connected switches. Consequently, the controller identifies the network topology (switches and communication links between them). Figure 4 depicts LLDP performance for discovering topology in the SDN network.

In [39], the researchers developed OFDPv2, an effective topology discovery method in OpenFlow-based SDN. It was developed due to the optimization of OFDP (Open Flow Discovery Protocol). The rationale behind OFDPv2 was to diminish the overhead of the topology discovery mechanism by decreasing the number of control messages the controller should transmit. Indeed, OFDP produces a particular LLDP packet for each switch port; it sends each network packet to the corresponding network switch through a special OpenFlow *Packet-In* message. Thanks to the OFDPv2 protocol, only a single LLDP packet is produced and transmitted to each network switch. The simulation results in [39] indicated that OFDPv2 notably reduced the control traffic overhead and the CPU load imposed on the SDN controller.

3.2 Traffic measurement

Effective network parameters in traffic should be evaluated for optimal engineering traffic, and network traffic should be analyzed and examined based on these parameters. Traffic measurement is one of the most crucial challenges for network management. Network managers can improve network management with dynamic and accurate traffic measurement and load balancing. In this section, the available methods of traffic measurement are investigated.

3.2.1 Measurement of effective parameters of network traffic

The purpose of establishing computer and telecommunication networks is to produce income. The effective network parameters should be evaluated in the design and operation states to achieve the network Service Level Agreement (SLA) and satisfy the users. If the impact of considered parameters in the designing stage fails to meet the QoS requirements of the network, these parameters should be rearranged and re-adjusted. For instance, we can adjust and configure network parameters to reduce network efficiency with redundant devices or communication links. For better traffic management in SDN networks, the respective parameters of the network should be precisely measured. These parameters include QoS, network topology discovery, and network traffic. Two active and passive methods measure massive traffic and diversity of SDN networks. In the dynamic measurement method, network traffic flows are continuously monitored; such monitoring is carried out by transmitting Probe packets to all the network paths for measuring one-way delay or round-trip delay.

In contrast, real-time network traffic is analyzed and investigated passively without transmitting the Probe packet. SDN includes two types of traffic: Data traffic, Control traffic. Control traffic refers to the traffic between controllers and OpenFlow switches, and data traffic attributes to traffic among the OpenFlow switches. In SDN, for determining the features of each flow, certain statistical information from the ports of each network switch, including the number of packets, size of network packets, and end-to-end traffic matrix for the whole network, should be collected.

3.2.2 Traffic measurement mechanisms in SDN

Several methods have been proposed for measuring and supervising network traffic in SDN. This section investigates and compares measurement mechanisms and how they operate. Various methods are proposed to monitor the traffic measurement mechanisms in SDN. In the following, we examine the proposed approaches.

OpenNetMon (OpenFlow Network Monitoring) has been proposed in [45] to measure the network performance with delay, efficiency, throughput, and packet delivery rate. This proposal is a module for monitoring the QoS in SDN. This module uses a pull-based method for monitoring.

iSTAMP has been offered in [46] to allocate the network resource, including CPU, TCAM, and bandwidth. This method divides the flows into smaller flows and manages the resource assignment. It uses a push-based method.
OpenTM (OpenFlow-Based Network Traffic Management) is another method to monitor the network query-based. It is used to estimate the network flow matrix [47]. This mechanism discovers flow paths and pulls Flow bytes and Packet-count counters alternately from flow path switches. Using routing information and collecting the statistics related to the productive flows from similar resources and their transmission to similar destinations, OpenTM creates a traffic matrix. It has overhead to poll switches for each flow path, while the random selection harms the precision.

PayLess is based on a flexible RESTful API for collecting statistics related to the flow with different accumulation levels [48]. PayLess collects real-time information with high accuracy without imposing overhead on the network. PayLess has more overhead and more accuracy. Nevertheless, this method has less overhead and more monitored data errors regarding greater time distance for polling.

The authors have proposed FlowSense in [49] to provide a monitoring module for the SDN controller that investigates and analyzes dynamic flows concerning the messages received by the controller. Also, it estimates the efficiency of the communication link of each flow using FlowSense, Packet-In, and Flow-Removed messages in OF networks. The results from the evaluation indicate that this method has higher accuracy and precision than polling-based methods.

DREAM (Dynamic Resource Allocation Measurement) is a management architecture approach for network hardware resources, which effectively balances resource cost and measurement precision. Resources are not allocated before measurement; instead, they are dynamically made available based on traffic features to achieve the appropriate precision level [50]. DREAM mechanism has three levels: user level, which is regarded as the high level, is responsible for measurement task; indeed, it has two responsibilities, i.e., determining task type and determining the threshold of traffic flow. The mid-level is the algorithm of the DREAM method, which can be executed in the SDN controller and receives tasks from the user; then, after creating task objects, it makes them available to the switches. SDN forwarding devices measure hardware storage resources at the lower level.

HONE (Host Network Traffic Management) is an approach proposed by [51] to do the required measurements; this mechanism uses software agents on hosts and a module that communicates with network devices. Since collecting statistics related to different flows is costly, HONE uses two procedures for analyzing statistical information related to flows. The first procedure, a lazy materialization of measurement data, uses data-based tables for uniform abstract representation of statistical data gathered from hosts. This procedure minimizes computational overhead, allowing the controller and host agents to analyze the required statistics requests for multiple task management. The second procedure provides parallel data transmission operators for programming data analysis logic. These operators are also used for accumulating the data gathered from hosts. One of the fundamental shortcomings of HONE is that it should be installed and it should be synchronized for processing queries from the statistical table.

A method named OpenSample has been worked on [52] and proposed through the investigations and experiments of the IBM recommended using measurement methods based on SDN sampling and Open Sample. Open Sample uses the sampling tool of the Sflow packet for capturing samples from the headers of the network packets with low overhead. Also, this method uses TCP serial numbers of the captured packet headers for measuring flow statistics.

The approach OpenSketch has been proposed in [53] and uses a measurement library in Control Plane for configuring and allocating resources for automatic measurement activities. OpenSketch provides a pipe construction line with three stages, i.e., hashing, filtering, and counting in the switch. It can be implemented by commodity switch components and supports many measurement activities. This three-stage pipeline is implemented on NetFPGA, such as an OF switch. OpenSketch library includes systems of sketches, sketch management, and a resource allocator. Sketches can be applied for measurement applications such as traffic change detection, flow size distribution estimation, and heavy hitters. Also, they can facilitate measurement in the SDN controller.

There is a method named PLANCK that facilitates available port mirroring in most commodity switches. Port mirroring is a more used solution for monitoring traffic passing through a mirror that uses network analysts. If this technique is used, packets will likely be lost due to the increased port traffic mass. To sort out this problem, buffering network packets can be used in the case of a traffic increase [54].

NetFlow is deemed a recognized system for analyzing and sampling packets that the Cisco company proposed. In NetFlow, the switch keeps the information related to each traffic in its cache and determines NetFlow of flows using Five-Tuples. NetFlow compares its header with the data saved in the local cache as soon as a flow arrives. In the case of compliance, it increases the number of packets; otherwise, it registers it in the cache table as a new flow [55].

OpenMeasure has been proposed as the smart sampling and network inference engine [56]. This module is located in the controller. This mechanism applies an online learning algorithm for specifying highly informative flows about sampling. The global view in SDN architecture could cause the resource monitoring and adjusting flow sampling in each switch. This module works based on OpenFlow API. This mechanism has three elements. The first element aims to detect the most informative flows from online learning and arranges rules that should be installed. The second element dynamically specifies where the rules should be installed throughout the network using the controller's global view. Ultimately, the third element periodically extracts traffic statistics from switches (TCAM counters, OF switches, and SNMP link loads) and uses the available inference methods for estimating traffic matrix or supporting other monitoring applications such as hierarchical heavy hitter identification.

DISTTM (Distributed Traffic Management) is a method based on the path selection in SDN, designed in the control plane [57]. This method supports the distributed topology to cause the controllers to collaborate. This approach is appropriate for inter-domain data center networks. DISTTM is based on four fairness, including (i) Fair Controller Distribution (FCD) is about fair requests distribution, (ii) Fair Domain Distribution (FDD) which refers to fairness in the identical distribution of flows, (iii) Fair Switch Distribution (FSD) that states fair load distribution among switches and (iv) Random (RD) which refers to fair controllers' distributions.

FleXam (Flexible Sampling Extension) is an approach to provide a flexible sampling development of OpenFlow, which makes it possible for the controller to access information at the packet level [58]. This mechanism has two.

kinds of sampling: (i) stochastic sampling which FleXam chooses those flow packets with ρ probability. (ii) Deterministic sampling refers to sampling where FleXam chooses m consecutive packets from each k successive packets and skips the initial δ packets.

OpenSAFE has been proposed in [59] to apply OpenFlow to monitor network traffic from security issues. This mechanism spanned network traffic to predetermined sinks based on the prespecified rules and strategies. OpenSAFE monitors hardware reserves, whereas network operators are unwilling to perform it.

DCM (Distributed & Collaborative Monitoring) in [60] refers to a memory-efficient distributed and collaborative per-flow monitoring mechanism. This method employs bloom filters for denoting monitoring rules using a small memory size. This method assumes that flows are frequently monitored redundantly at different switches if flow aggregation decreases flow rules. Furthermore, if single flows are chosen to obtain fine granular measurements, the quantity of regulations becomes overwhelmingly large. The proponents of DCM dealt with this issue by utilizing two-stage bloom filters on switches. Filters might be specified, so network switches monitor specific flows regardless of defining one rule per flow. As a result, rules can be effectively determined in harmony with the monitoring rules of other network switches. That is, DCM facilitates collaboration on the switch level.

DISCO (Distributed SDN Controllers) has been proposed in [61]**,** which announces monitoring agents in controllers to measure link utilization. Since traffic monitoring in DISCO is restricted to the links between peering spots in neighbouring networks and since the controller individually measures statistics, collaboration does not occur in this domain. In DISCO, the control plane lets controllers maintain communications with each other. Thus, DISCO might be regarded as an appropriate foundation for meeting the requirement of establishing communications between controllers.

The Path-Mon is another approach discussed in [62] that converts flow and path information into tags; such tags are used to create the required flexibility and corresponding link-to-link correlations for detecting anomaly one or doing traffic engineering operations. Network managers can query flow statistics at different aggregation levels. Since monitoring entries are precisely compatible with flows of interest, unrelated flows are removed from the statistics announced by the switches.

The authors have proposed Flo-V in [63] to fulfill network monitoring in Virtualized SDNs (vSDNs). Flo-v produces precise monitoring information for a vSDN. It uses selective and adaptive techniques for monitoring traffic in a network hypervisor, and the created overhead in this method is less than those of other methods. Simulating this method indicated that it could obtain useful network information regarding minimal resource use regarding network and CPU utilization.

LiteFlow suggests an intelligent mechanism for selecting an authority switch to monitor all flows between an end-host pair; however, the other path switches forward packets regardless of monitoring processes [64]. LiteFlow allocates and distributes monitoring flows among SDN switches, correctly handles scalability, and enables accurate network monitoring. This method includes two distinct modules, i.e., Flow Partitioner and FlowMon. Flow Partitioner aims to distribute and share monitoring among switches and optimize their resources. Moreover, FlowMon is regarded as a kind of application that implements numerous network metrics by using Flow Partitioner as a basis.

MDCP (Measurement-Aware Distributed Controller Placement) tried to modify the paradigm of software-defined measurement [65]. This method was aimed at fundamentally improving measurement overhead. In SDN, randomly placing the number of controllers increases communication costs and reduces performance. It should be noted that the design of controllers remarkably affects measurement overhead. According to this interesting finding, the proponents of MDCP suggested that if controllers are appropriately placed, measurement overhead in SDN can be significantly diminished.

COSTA (Cross-layer Optimization for Sketch-based Software Defined Measurement Task Assignment) denotes a novel method that made a trade-off between task accuracy and measurement performance [66]. The cross-layer way expressed this issue in the mixed-integer nonlinear programming problem. The summary of the reviewed papers is presented in Table 1.

3.3 Network traffic load balancing

Traffic in an SDN-based network includes (i) Traffic data plane and (ii) Traffic control plane. The integrated management of the controller in SDN networks is regarded as a merit for load balancing [67]. In other words, strategic planning for maintaining load balance in SDN networks is easily feasible, which might not be the case in traditional networks. On the other hand, it should be mentioned that load balancing in SDN networks faces particular challenges [68]. For instance, ECMP is one of the multi-path routing protocols; it receives a packet and investigates its header for measuring hash. It selects multiple paths based on hash value and transmits the packet to that path. In this protocol, packets with similar IPs are sent in similar ways because the hash value of all of them is identical.

Consequently, heavy SDN traffics, referred to as elephant flows, will be transmitted from one side of the path, violating load balancing. On the other hand, as soon as OF switches in SDN networks receive a flow, if this flow is not compatible with any available rules in the network switch, the first packet of that flow will be transmitted to the controller for decision-making. The controller launches a new forwarding rule based on the received packet in the switch. The new forwarding rule installation process is time-consuming for the traffic and will cause plenty of delays. Hence, load balancing techniques are developed for the data plane and control plane to prevent additional overheads in the data plane and control plane. The following section investigates the available load balancing methods for the data and control planes.

3.3.1 Data plane load balancing

As mentioned earlier, one of the shortcomings of ECMP is that elephant flows with identical hash values for their headers are transmitted through the same path. This operation violates load balancing and produces excellent overhead. Specific methods such as Hedera [69], Mahout, and MicroTE have been proposed for addressing this issue.

Authors have proposed Hedera, a centralized, dynamic, and scalable traffic management method that schedules packet flow to enhance network resource efficiency [69]. This mechanism is introduced to make the data centers effective and fair network bandwidth. Consequently, traffic scheduling will lead to optimal bandwidth allocation for packet traffic and fairness in the network. Thus, load balancing will be achieved. To fulfil these objectives, the Hedera method collects the statistics of flows from switches every five seconds to detect large flows. As a result, traffic needs, routing flows and determining non-conflicting routes do not allow ECMP protocol in this method. The Hedera method determines and manages significant traffic and elephant flows to balance load and optimal use of multiple paths between data centers. Strategic scheduling of Hedera includes: (i) Specifying elephant flows in edge switches, (ii) Alternately collecting information of flows from switches and determining elephant flows in case the sizes of flows are more significant than the threshold value. (iii) Determining service requirements for elephant flows, such as allocating appropriate bandwidth.

Authors in [70] have worked on a suitable method for managing and optimizing traffic for SDN networks. This mechanism changes the end-hosts to detect and manage elephant flows. Traffic management cost is reduced by using an additional back-end service. Compared with Hedera, Mahout reduces traffic management costs and improves management efficiency, attributed to using a back-end server instead of forwarding equipment to determine elephant flows. Earlier methods alternately inspected traffic statistics of switches for selecting elephant traffic. Also, previous methods used

Table 1 Summary of traffic measurement mechanisms in SDN

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[45]	OpenNetMon	✓	–	–	✓	Adaptive switch data fetching	High	Low	–	–	–
[46]	iSTAMP	✓	✓	✓		TCAM partitioning	High	Low	–	–	–
[47]	OpenTM	✓	–	✓	–	Polling the switches to gather flow statistics	High	High	–	–	–
[48]	PayLess	✓	–	–	✓	Adaptive polling based on a variable frequency flow statistics collection	High	High	–	–	Low
[49]	FlowSense	✓	–	✓	–	Path utilization Calculation based on OF message	High	Low	–	–	–
[50]	DREAM	–	✓	–	✓	Dynamic resource allocation to improve accuracy	High	–	–	–	–
[51]	HONE	✓	–	✓	–	Using the agent to gather data from hosts and FEs	–	–	Low	–	–
[52]	OpenSample	✓	–	✓	–	Using packet sampling with sFlow and TCP sequence number to detect elephant flows	High	–	–	High	–
[53]	OpenSketch	–	✓	–	✓	A hierarchical heavy hitter scheme to reduce the overhead of monitoring	High	Low	–	Low	–
[54]	PLANCK	–	✓	–	✓	Port monitoring	–	Low	–	High	–
[55]	NetFlow	–	✓	–	✓	Using cash for Netflow information storing	–	High	–	High	–

Table 1 (continued)

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[56]	OpenMeasure	✔	–	–	✔	Online learning is used to predict and update the measurement rules dynamically	High	Low	–	–	–
[57]	DISTTM	–	✔	✔	–	Reduces monitoring overhead via collaboration between controllers	–	Low	High	–	–
[58]	FleXam	–	✔	✔	–	A flexible sampling with OF with access to the packet–level information for monitoring and security applications	High	Low	–	–	–
[59]	OpenSAFE	✔	–	✔	–	Using programmable fabric to collect statistics of network and detect malicious activity	High	Low	–	–	–
[60]	DCM	–	✔	✔	–	Data plane modification to customize the protocol	High	Low	–	–	–
[61]	DISCO	–	✔	✔	–	A model for managing the domain and communicating with other controllers to provide end–to–end net service	High	Low	–	–	–
[62]	PathMon	–	✔	✔	–	Tagging the flow based on flow information and encoding for TE	High	Low	–	–	–

Table 1 (continued)

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[63]	Flo–v	–	✓	✓	–	A selective and adaptive method for TE in the hypervisor	High	Low	–	–	–
[64]	LiteFlow	–	✓	✓	–	To balance the load by distributing it among SDN switches	High	Low	High	–	–
[65]	MDCP	–	✓	✓	–	Formulates the measurement–aware distributed controller placement problem as a quadratic integer programming problem	High	Low	–	–	–
[66]	COSTA	–	✓	✓	–	Cross–layer information management to solve a mixed–integer nonlinear programming problem	High	Low	–	–	–

a sampling of the packets of switches with high overhead, increased monitoring time, and the consumption of network resources.

For enhancing network efficiency and reducing overhead determining elephant flows, instead of direct monitoring of network switches, Mahout operates through the shim layer of the operating system. This layer is responsible for monitoring the local traffic through the socket buffer. Mahout determines it as an elephant flow when the buffer surpasses a certain threshold and marks its packets. Mahout considers priorities for the rules of the table of flows: high priority and low priority. According to these prioritizations, ECMP is used for transmitting packets compatible with low priority. The packets of an elephant flow compatible with high priority rules are sent to Mahout for measuring the appropriate path, and the transmission rules are updated in the switch. By doing so, the flow bottleneck is transmitted from the network layer to the application layer.

MicroTE is an approach that refers to fine-grained traffic engineering that can be applied in the data center networks topology and uses an end-host elephant flow detection scheme to detect large flows (elephant flows) [71]. Instead of allowing the controller to investigate and check switches alternately and directly, MicroTE has a monitoring component on the server-side. MicroTE reacts to changes in network traffic and scales down an extensive network, reducing processing overhead. MicroTE has the following advantages: (i) In the case of fundamental changes in network traffic, it allows the controllers to update the realized changes in network traffic. (ii) Through investigating and constant checking of switches at any second and gathering statistical traffic information, controllers cause a significant increase in network traffic load, but MicroTE prevents controllers from doing so. (iii) Bottleneck transfers the traffic load from switches to hosts. Each of the servers in the Monitoring Component monitors input and output data traffic on its ports, and only one server in each rack is responsible for aggregating traffic packets. These servers aim to collect data from other servers of each server rack, aggregate server-to-server data to rack-to-rack data, and determine a switch for transmitting aggregated data and transmitting them to the network controller.

A load-balancing method called DevoFlow is based on the technology of Wildcard scheduling, which aims to reduce the number of exchanges between controllers and switches [72]. Due to wildcard rules, the network switches can route microflows locally, and the controller focuses on the elephant flows that require QoS provisioning. In sum, it should be mentioned that DevoFlows' scheduling microflows and elephant flows are based on switch and controller, respectively. Accordingly, scalability and meeting QoS are feasible for different flows.

MiceTrap is a scalable traffic engineering scheme that can detect the Mice-flows (short-lived flows) of the data center and uses the OpenFlow group table to aggregate the incoming Mice-flows for each destination with a weighted routing [73]. This weighted routing algorithm aims to find a set of dynamically computed ratios (weights) used to spread the traffic at each hop across the available next hops for given traffic demands. MiceTrap achieves load balancing by spreading the incoming Mice-flows across multiple paths from source to destination, considering the current network load.

DIFANE is the algorithm that is regarded as another traffic scheduling method based on Wildcards which has two fundamental principles: (i) Controller implements some rules in a set of virtual switches. (ii) Switches schedule all packets in the data plane[74]. If traffic flows are incompatible with the cached rules of ingress switches, these switches will encapsulate them; then, based on specific information, authority switches investigate packets in Data Plane and transmit feedback ingress switch for locally saving the related rules. DIFANE architecture includes a controller that establishes rules and applies them to authority switches. Authority rules have higher memory and processing power than other switches. DIFANE has three sets of rules: (i) Cache rules: these rules are concerned with ingress switches installed on them by authority switches. Ingress switches manage the majority of data traffic. (ii) Authority rules: the controller installs, launches, and updates these rules on authority switches. If a packet is compatible with this set of rules, the controller installs a series of rules in ingress switches. (iii) Partition rules are installed by the controller in all switches. Given all these rules, at least one packet is compatible with rules and remains in the data plan.

A distributed load balancing algorithm was proposed in [75] to dynamically balance control traffic across a cluster of SDN controllers named Wardrop. Hence, it reduces latency and enhances the entire cluster throughput. This algorithm was designed according to the game theory, which moves towards a certain equilibrium called Wardrop equilibrium. In other words, this algorithm is intended to dynamically learn the best efficient amalgamation of flow rates from each network switch to the available controllers. Thus, it can be argued that this method enhances the total throughput and reduces control connection latency.

The proponents of NetAlytics intended to investigate how SDN, NFV, and big data analytic techniques may be combined for robust monitoring and debugging mechanism about distribution systems called NetAlytics [76]. It makes it possible for the network supervisor to determine a straightforward query responsible for specifying different types of traffic to be monitored and data to be collected. Furthermore, it determines how data should be analyzed. The query is converted to an array of SDN rules, leading the favourite traffic to dynamically instantiated NFV monitors that efficiently extract the target data. Then, the real-time data is accumulated and transmitted via a scalable streaming analytics engine, making it possible for system administrators to obtain meaningful understandings of their networks and applications quickly.

SOTE (Software OSPF-Based Traffic Engineering) is a TE method developed as a hybrid traffic engineering mechanism that combines OSPF and SDN to reduce link consumption in the network [77]. The proposed model can alert OSPF messages and flow distribution among nodes in SDN. The rationale behind this method was to regulate the weight setting of the whole network for balancing flows that come out of regular nodes of the network. It was also aimed at splitting flows that aggregate at the SDN nodes to reduce the maximum link utilization of the whole network.

This is a congestion-aware and multipath-based forwarding traffic engineering scheme for SDN. MSDN-TE (Multipath-based SDN Traffic Engineering) dynamically forwards incoming traffic to the best selected shortest paths in the network. This TE mechanism is the extension of the OpenDayLite controller [78]. This scheme gathers network state information and considers the actual paths load to forward the incoming flows on available multiple selected shortest paths.

A Deep Learning-Based System has been proposed to increase the network performance and avoid traffic congestion in SDN [79]. This method detected various flows to distribute the identified traffic to multiple queues with different priorities. It also shaped the traffic to manage the bandwidth and incoming flows. It has been implemented to consider the port capacity to accomplish general load balancing.

An efficient traffic management solution in data center networking has been proposed to provide the necessary network resource based on SDN demands [80]. The proposed model has worked on collaborating the online routing method with the multi-path transmission control protocol (MPTCP) and segment routing (SR) in a software-defined network for better results in DCN.

An efficient approach has been suggested to deliver multimedia content by solving multicast routing as a delay constraint least cost (DCLC) problem [81]. DCLC is an NP-Complete problem; therefore, teaching–learning optimization has been used to solve it.

A traffic engineering-aware distributed routing (TEDR) has been proposed to minimize maximum link utilization as the TE objective and comply with SDN waypoint enforcement and TCAM resource limitations [82]. The authors formulated the TE problem as an integer linear programming (ILP) and solved it centralized for SDN waypoint selection and splitting fractions for each flow. They have used a distributed algorithm deriving from Lagrangian decomposition theory to solve the TE problem effectively.

To make the traffic engineering scalable for efficient mapping traffic demands to paths in SDN datacenters, an optimal solution has been proposed to schedule elephant flows across paths to mitigate network congestion and improve load balancing in SDN [83]. The authors expressed that this method is still not applicable because the needed time for optimal path calculation is not tolerable in SDN; hence, they had to limit the search space based on the linear programming method to solve this problem reasonably.

The following summary of load balancing in the data plane is stated in Table 2.

3.3.2 Control plane load balancing

The daily traffic increases in SDN data centers [15] and the centrality of the controller in these networks, designing and using several controllers can eliminate the communication bottleneck between OF switches and controllers. Some of the solutions proposed for sorting out bottleneck problems are given below.

Hyperflow is considered an event-based distributed controller that uses OF protocols for configuring switches; therefore, minimizing response time, Hyperflow assigns decision-making to some controllers [84]. All controllers have a similar view of the network through synchronization methods and locally give services to requests without calling other nodes and asking for their help. Hyper flow-based networks include OF switches for transmitting data, Nox controllers for decision-making and services for Data plane requests, and an event distributing system for cross-layer communication. Switches communicate with one of the nearby controllers to prevent congestion. All controllers have a compatible and consistent view of the entire network.

Onix is a distributed controller executed in one of the clusters consisting of physical servers [85]. Network control logic is installed above Onix API to survey the appropriate behaviour of the network. Onix core is considered a suitable and general API for controlling the infrastructure network, making it possible to develop new applications. Onix saves a replica of network topology in a table known as NIB (Network Information Base); it can provide scalability and reliability in the network by replacing and distributing NIB data between multiple controllers being executed.

Balanceflow is a controller with an architecture capable of load balancing, dividing traffic control load among other controllers [86]. It operates when some controllers' load is high; it transmits some of the load to other controllers with less load to balance the load. This way, efficiency is enhanced, and the network effect is reduced. Different flow requests of each switch are dedicated to other controllers. Balance flow controllers save the information related to its flow requests and alternately publishes this information through a cross-layer communication system.

Table 2 Load balancing in the data plane

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[69, 45]	Hedera	–	✓	–	✓	In the network, it detects elephant flows to reduce the overhead	–	Low	–	High	–
[70]	Mahout	–	✓	✓	–	It discriminates the elephant flows and informs the controller by an added layer named ’shim’	–	High	–	High	–
[71]	MicroTE	✓	–	✓	–	This method is implemented in a dependent machine to detect elephant flows proactively	–	Low	–	High	–
[72]	DevoFlow	–	✓	✓	–	Reduces the controller overhead by assigning a set of wildcard rules to OF switches	–	Low	–	–	–
[73]	MiceTrap	✓	–	✓	–	It achieves load balancing by spreading the incoming Mice–flows across multiple paths from the source to the destination	–	Low	–	High	–

Table 2 (continued)

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[74]	DIFANE	–	✔	✔	–	Introduces authority switches in the network to handle basic flows	–	High	–	High	–
[75]	Wardrop		✔	–	✔	It presents a distributed load balancing algorithm dynamical balancing the control traffic across a cluster of SDN controllers	–	–	–	High	–
[76]	NetAlytics	✔		✔	–	This approach deploys customized network monitors transparent to end–host applications and leverages a real–time extensive data framework to analyze application behaviour in a time–consuming manner	–	Low	High	High	–
[77]	SOTE	✔	–	–	✔	It uses OSPF and SDN to increase link efficiency	–	–	–	High	Low
[78]	MSDN–TE	–	✔	✔	–	It gathers network state information and considers network path load in forwarding the flows to available					

Table 2 (continued)

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
multi–path	–	High	–	High	–						
[79]	DeepLearning Based TE	✓	–	–	✓	It uses a deep learning method to avoid congestion with shaping. This method makes the network load balanced	High	–	–	High	–
[80]	MPTCP + SR		✓	✓	–	It proposed a routing method in the data center to optimize the network resources	–	High	–	High	–
[81]	Teaching–Learning Optimization	–	✓	✓	–	Solve DCLC problem using teaching–learning optimization	–	High	–	High	–
[82]	TEDR	–	✓	✓	–	A distributed algorithm for solving the TE optimization problem	–	Low	–	High	–
[83]	Elephant Flow scheduling	✓	–	✓	–	An optimization based on flow demands mapping to paths in datacenters	–	Low	–	High	–

Consequently, load balancing is appropriately achieved. Balance flow has two controllers: super controllers are ordinary controllers. Super-controller is responsible for establishing load balance for all the controllers. When the average number of flow requests exceeds the threshold of the flow-request rate, load balancing in the network is not achieved. Hence, the super-controller begins to balance the load. The point can be set based on the super controller efficiency: the number of controllers and the network environment.

Kandoo is a hierarchical controller operating at two levels: local and logically centralized root controllers [87]. The local level of the controllers execute requests related to local operations since they are close to the switch. Second-level controllers perform non-local functions, i.e., those operations at the global level, and require comprehensive information from the infrastructure network. Kandoo controller has several local controllers and a logically centralized root controller. In Kandoo, each switch is controlled by a Kandoo controller, and each Kandoo controller can manage several network switches.

LBDC (Load Balancing problem for Developed Controllers) explores the usage of developed SDN multiple controllers to monitor, manage, and coordinate mega data centers [88]. This scheme presents new techniques to overcome unbalanced workload in controllers and reconfiguration complexities. The authors design multiple solutions for LBDC, including linear programming with rounding approximation, three centralized greedy algorithms, and one distributed greedy algorithm. LBDC improves scalability and increases availability, throughput, and performance.

LBBSRT (Load Balancing Scheme Based on Server Response Time) is an efficient SDN Load Balancing method [68]. Using the real-time response time of each server measured by the controller for load balancing operation, this scheme processes user requests by obtaining evenly balanced server loads. LBBSRT has low cost and improves the system reliability and scalability.

A traffic engineering technology based on segment routing in SDN has proposed a novel method called SRTE-L [89]. The authors have submitted to set a path decision variable L and a path constraint to limit the length of each segment routing (SR) path and the number of intermediate nodes, thereby minimizing the maximum link utilization and reducing computation time. The authors investigate the trade-off between link utilization and computation time.

The brief of this review about load balancing in the control plane is presented in Table 3.

3.4 Quality of service

Quality of Service is an issue that has risen nowadays with the advent of varied applications, including VoIP, IoT, e-commerce, cloud computing, video conferencing, and online gaming. Different types of networks like MANET (Mobile Ad hoc Network), VANET (Vehicular Ad hoc Network), and WSN have caused significant challenges in providing the required QoS. The essential metrics in QoS are end-to-end delay, packet loss, and assuring bandwidth. From the network point of view, service quality refers to the fact that the network should behave appropriately with packets and flows based on their needs. At the packet level, end-to-end delay, jitter, and packet loss are three parameters used to define service quality. This section addresses the approaches which had been proposed for QoS metrics improvement.

The greedy algorithm has been used for each flow to guarantee service level in SDN [90]. A.Shirmarz et al. have worked on adaptive resource allocation to improve the QoS of each application. They have used the greedy algorithm to sort the paths between each couple of nodes. This method could simultaneously improve the network performance and QoS because their approach has considered the trade-off between the network resource and application requirements.

TOPSIS algorithm has been proposed for the best path selection in SDN [91]. A. Shirmarz et al. have used the controller as the central brain of the network in SDN and proposed the Topsis algorithm to improve performance networks automatically. This method has used different performance metrics to solve decision-making, making the QoS better than similar approaches.

DTE (Dynamic Traffic Engineering) implementation in Software Defined Data Center Networks has been proposed [92]. This algorithm simultaneously selects the best path to improve QoS and user satisfaction. This approach could improve the QoS from throughput, delay, and jitter points.

DynamiTE (Dynamic Traffic Engineering) in software defined cyber-physical systems have been proposed to minimize the control overhead at the SDN controller by minimizing the number of PACKET-IN messages [93]. A greedy heuristic approach is used for ternary content-addressable memory (TCAM) using space optimization that causes throughput in SDN.

A novel flow routing algorithm based on non-dominated ranking and crowd distance sorting to improve SDN performance has been proposed by A.Shirmarz et al. [94]. The model has been used for network performance improvement using the multi-paths routing that causes the QoS amelioration.

A heuristic traffic engineering approach is used in SDN to switch and forward the flows through the multi-path situation [95]. The proposed dynamic path selection has two critical priorities: path cost and load. This method selects the path based on the flow request to provide the required QoS. This approach can improve the QoS of applications with proper bandwidth allocation.

A thesis about application-aware traffic engineering in SDN [96] guides the traffic through the path, which is proper

Table 3 Load balancing in the control plane

Reference	Proposed Approach	Architecture		Monitoring		Method	Optimization Metrics				
		Centralized	Distributed	Push	Pull		Accuracy	Overhead	Scalability	Performance	Cost
[84]	Hyperflow	–	✔	✔	–	The Publish–subscribe method with file system for cross–controller communication and global view. Distribution of controllers and directing requests to the nearest controller	–	High	High	–	–
[85]	Onix	–	✔	✔	–	This method uses the Publish–subscribe method with the NIB database system. Onix is a platform that runs on a cluster of one or more physical servers	–	High	High	High	–
[86]	BalanceFlow	–	✔	✔	–	A super controller and many normal ones exist. The super one is responsible for balancing the controllers	–	High	–	–	Low
[87]	Kandoo	–	✔	✔	–	The controllers' distribution has been used to handle network traffics.Local controllers execute local applications by controlling one or some switches. The root controller controls all local controllers	–	Low	Low	–	–
[88]	LBDC	–	✔	✔	–	It presents new schemes to overcome unbalanced workload among controllers and reconfiguration complexities	–	Low	High	High	–
[89]	SRTE–L	–	✔	–	✔	It proposed a routing to minimize the length of each segment routing path and the number of intermediate nodes	–	Low	–	High	–
[68]	LBBSRT	–	✔	✔	–	It processes user requests by obtaining evenly balanced server loads	–	–	High	High	Low

for each application type. The idea has proposed a method for resource allocation for QoS provisioning in different applications.

HiQoS (High Quality of Service) guarantees QoS in SDN networks [97]. HiQoS specifies multiple paths between the source node and the destination node and, using a queuing mechanism, provides QoS parameters for different types of traffic. HiQoS uses ECMP protocol for finding multiple paths. In this way, this method uses numerous paths for determining the priority for each of the traffics and allocates the appropriate queue. Consequently, it can meet the requirements related to bandwidth and delay for each of the flows. On the other hand, HiQoS can afford reliability by using multiple paths because it can transmit packets through different approaches if one of the paths fails.

OpenQoS method is an appropriate controller for satisfying QoS to support video flows [98]. Using packet headers, OpenQoS can transmit network flows via two multimedia flows capable of meeting QoS requirements, and data flows are transmitted through best-effort paths. Appropriate paths are selected based on the delay and packet loss parameters.

F. Ongaro et al. have proposed a new method in [99]. They developed an integrated traffic scheduling method based on QoS for SDNs. This framework faces the following problems and challenges: (i) regarding networks with a shared node or path, how can a suitable path be selected for commercial flows of multiple applications? (2) meeting QoS requirements, the total allocated bandwidth for each path should not exceed the capacity of the physical link. To overcome these problems, this framework has some key components, which include the following modules: (i) module for mapping network topology, (ii) module for collecting network status, (iii) module for selecting the path, and (iv) module of configuring dynamic path. The first two modules monitor the data plane to update topology, collect the dynamicity of network parameters and produce weights for each path in the network diagram. The path selection module selects appropriate paths based on QoS requirements and the weights of the network diagram. The module configures dynamic paths and is responsible for updating routing rules in the data layer at appropriate times. As the chief module, the path selection module decides about scheduling using for paths, the MCFCSP model (Multi-Commodity Flow and Constrained Shortest Path).

iMOS (Intermediate Mean Opinion Score) method offers a procedure for advanced monitoring of VoIP in SDNs. OpenFlow was used to execute iMOS quality indicators to understand and perceive per-hop VoIP quality [100].

SDN-MPLS has been proposed for routing in the mobile network to provide the required QoS [101]. The authors have tried to make a trade-off between network load balancing, route length, and energy-saving with low complexity in mobile networks.

Intelligent Traffic Control (ITC) has been suggested to improve delay, jitter, and throughput using deep reinforcement learning for routing [102]. This paper has addressed the QoS in hybrid SDN.

A proposed machine learning-based approach for QoS & QoE improvement in SDN[103]. This approach has used the tagged data set for the training model to improve the QoS metrics like delay, jitter, packet loss & throughput.

A classical Shuffled Frog Leaping Algorithm (SFLA) has optimized energy consumption with QoS constraints [104]. The proposed routing has provided the required QoS, including delay, jitter, and packet loss.

Multi-layer slicing and resource allocation for SDN/NFV 5G to ensure QoS metrics have been suggested in [105]. The traffic QoS requirements have been provisioned with a QoS-aware TE method.

A control framework for virtual provisioning in an SDN-based Internet service provider (ISP) network has been proposed and used a heuristic TE algorithm in polynomial time [106]. The configurations have been changed to provide Scalable, Robust, and QoS-Aware Virtual-Link Provisioning in SDN.

A QoS-aware traffic engineering method in SDN has been suggested for improving routing delay and packet loss ratio [107]. This path calculation increases the computational time in the controller that QoS-Aware routing has reduced the load with network resource utilization improvement.

DTE_SDN has been used to schedule the delay-sensitive traffic with QoS metrics (throughput and delay) monitoring for QoS improvement [108]. The dynamic DTE-SDN scheduling improves link delay and throughput as the QoS metrics.

Dynamic bandwidth allocation based on QoS demand in SDN has been proposed in [109]. The end-to-end dynamic bandwidth allocation causes the throughput, delay, and packet loss to improve.

The intent-based optical transport network infrastructure using an intelligent TE algorithm based on SDN and optical label switching (OLS) has been proposed to provide QoS parameters according to user intention during peak hours [110]. Intent-Based Software-Defined Transport Network (IBSDTN) states based on ML algorithms k-means and c-means.

Deep Reinforcement Learning (DRL) has been used to achieve an efficient network control scheme for TE called scaleDRL [111]. The authors used pinning control theory to select a subset of links in the network and name them critical links. They used the network information gathered by the controller to score the links as weight dynamically.

TEL is a system including two fast re-routing algorithms (FRR) used in [112] TEL-C and TEL-D, respectively. TEL-C computes backup forwarding rules in the control plane and tries max–min fair allocation. TEL-D is the mechanism to provide FRR in the data plane. These mechanisms minimize the memory on programmable data planes.

A hybrid MPLS-VPN (Multiprotocol Label Switching-Virtual Private Network) technique has been proposed for QoS provisioning [113]. The method tries to transfer data and voice in a trouble-free environment simultaneously. The proposed method improves QoS using traffic engineering.

This section shows the QoS parameters, including bandwidth, delay, jitter, and packet loss, for each proposed solution in Table 4.

3.5 Dependability

This section addresses the papers that have worked on the software-defined networks' dependability using traffic engineering. Three significant goals have been defined for dependability, including reliability, availability, and performance.

LBBSRT (Load Balancing Scheme Based on Server Response Time) is an efficient SDN Load Balancing method that makes the network reliable against any failure and congestion [68]. This method improves network reliability and availability.

HiQoS (High Quality of Service) is an approach to provide network reliability while it provides QoS metrics like bandwidth, delay, and jitter [97].

B4 is SDN-based WAN, which Google developed for establishing communications among data center networks of this company at different locations of the world [114]. The main goal of B4 is to resolve the dependability parameters such as reliability, performance, and fault tolerance. In particular, B4 simultaneously supports standard routing protocols and centralized traffic engineering for managing switches. Using traffic engineering, B4 can control the network edge to adjudicate among competing demands during resource constraints. It also uses multi-path forwarding or tunnelling to leverage available network capacity according to application priority and dynamically reallocates bandwidth in the face of link/switch failures. B4 architecture consists of three layers: (i) global layer, (ii) site controller layer, and (iii) switch hardware layer.

SWAN (Software-driven Wide Area Network) is a TE mechanism proposed by Microsoft company that can be applied for inter-data center WANs [115]. SWAN improves the utilization of these systems by centrally controlling and re-configuring the data plane to match current traffic demand. This TE mechanism achieves high efficiency while meeting policy goals such as preferential treatment for higher-priority services and fairness among similar services. This high efficiency of SWAN can be obtained via frequent network updates, globally coordinating the sending rates of services, and centrally allocating network paths. These updates can be implemented quickly without congestion or disruption by leaving a small scratch capacity on the links and switching rule memory.

To prevent the network from failing against the link loss, an efficient and survival traffic engineering (EFSUTE) has been proposed based on a survivable software-based TE model over SDN to increase reliability in real-time intelligent environments (IE) [116]. EFSUTE has used the abilities of SDN to compute and install two disjoint paths between any source and destination pair in the network.

Research on improved traffic engineering fault-tolerant routing in software defined wide area networks (SD-WAN) has been done [117]. The proposed mathematical model formalized SD-WAN data plane construction to switch the access network to more than one border router to improve fault tolerance. The authors used a virtual gateway to implement this model.

For real-time traffic, a smart routing system was proposed for a fault-tolerant reactive routing model in a software-defined wide area network (SD-WAN) [117]. The system can provide high availability and reliability in SD-WAN against routers' failure.

Secure-based traffic management has been proposed in [118] to improve network availability and reliability. The authors have suggested a mathematical model as classical TE. The flows are guided by SDN's network load, bandwidth, and reliability conditions. The controller redirects the flows towards the reliable path to increase reliability and availability.

4 Open issues for future research

Given the novelty of SDN in academic and industrial environments, there are undoubtedly several challenges in research on SDN-based traffic engineering that should be addressed in future studies. Some of these related research lacunas are mentioned and underscored in the following:

1. Dynamic load balancing in SDN-based traffic engineering: comprehensive technologies and novel applications such as 5G, CPS (cyber-physical system). Load balancing in SDN to the available network status can be considered a considerable challenge that should be systematically addressed in future studies. It should be noticed that the lack of dynamic load balancing in the network can increase packet loss. Hence, detecting huge traffics, i.e., elephant traffics and small traffics, namely mouse traffic and the appropriate bandwidth allocation

Table 4 QoS Parameters of each proposed solution

Reference	Proposed Scheme	Solution	Bandwidth	Delay	Jitter	Packet loss
[90]	Adaptive Greedy Routing	A greedy heuristic method for the path ranking improves each flow's QoS	✔	✔	✔	✔
[91]	TOPSIS Algorithm	The TOPSIS decision–making algorithm is proposed to select the best Quality for each flow	✔	✔	✔	✔
[92]	DTE	A Dynamic Traffic Engineering method for QoS guarantee in SDN	✔	✔	✔	–
[93]	DynamiTE	Decrease the number of Packet-In messages and minimize the TCAM used space in SDN	✔	–	–	✔
[94]	A routing algorithm based on non-dominated ranking and crowd distance sorting	Flow routing to improve the QoS using non-dominated ranking and crowd distance sorting	✔	✔	✔	✔
[95]	Heuristic TE	A heuristic traffic engineering approach to select the best path with low cost & low load to provide the flow QoS requirements	✔	–	–	–
[96]	Application-Aware TE	The TE is based on application resource requirements. Different applications have varied flows and conditions that are provisioned with the controller	✔	✔	✔	✔
[97]	HiQoS	Multi-path routing and queuing scheme for multimedia flow applications	✔	–	✔	✔
[98]	OpenQoS	A controller design for QoS-enabled routing scheme for multimedia traffic transmission	✔	✔	✔	–
[99]	MCFSP	It develops as an integrated traffic scheduling method based on QoS for SDN	✔	✔	–	✔
[100]	iMOS	It offers a procedure for advanced monitoring of VoIP in SDN	✔	✔	-	✔
[101]	SDN-MPLS	MPLS & SDN for routing performance improvement for QoS provisioning	✔	✔	–	–
[102]	Intelligent Traffic Control	Deep reinforcement routing for hybrid SDN QoS improvement	✔	✔	✔	✔
[103]	Machine learning-based WAN	Machine learning testbed for QoS & QoE measuring in SD-WAN	✔	✔	✔	✔
[104]	SFLA	A routing for energy consumption optimization and provide the QoS as a constraint	–	✔	✔	✔
[105]	Multi-layer slicing and resource allocation	Multi-layer slicing and resource allocation for SDN/NFV 5G to ensure QoS metrics	✔	✔	–	–
[106]	QoS Control Framework	A framework for configuration management in SDN for robust, scalability, and QoS virtual link	✔	✔	✔	✔
[107]	A QoS-aware traffic engineering	A QoS-aware TE for computational load with resource utilization improvement	✔	✔	–	✔
[108]	DTE-SDN	Dynamic scheduling with DTE-SDN to improve link delay and throughput	✔	✔	–	–
[109]	End-to-End dynamic bandwidth allocation	The end-to-end dynamic bandwidth allocation for throughput, delay, and packet loss as QoS metrics	✔	✔	–	✔
[110]	IBSDTN	ML algorithms k-means and c-means	✔	✔	✔	✔

Table 4 (continued)

Reference	Proposed Scheme	Solution	Bandwidth	Delay	Jitter	Packet loss
[111]	DRL	Deep reinforcement learning for dynamic link weighting for QoS improvement	✔	✔	✔	✔
[112]	TEL	TEL-C & TEL-D for control and data plane optimization	✔	✔	✔	✔
[113]	MPLS-VPN	MPLS-VPN to transfer data & video simultaneously	✔	✔	✔	✔

for each, are important challenges in SDN-based traffic engineering that must be investigated in further research.

1. TE in SDN-based wireless and mobile networks: in line with the increasing development of wireless and mobile networks such as VANET (vehicular ad-hoc networks) [119], MANETs (mobile ad-hoc networks), WSN, cellular networks (including 4G and 5G) [120] and the policy of being able to access information at any time and location, SDN and traffic engineering can play significant roles in these networks as well as wired networks [121]122, 123, 124, 125]. SDN and traffic engineering can reduce operational and management costs in these networks via heterogeneous technology, optimize their performances and support their various services. Thus, there is a pressing need for further research on these issues by interested academic and industrial researchers.
2. Multiple controllers Synchronization in SDN: in some networks like IoT, big data, and cloud computing, several controllers control the FEs, resulting in network scalability and fault tolerance [126]127]. Synchronization and consistency of these controllers in operations such as topology discovery are fundamental challenges in SDN-based traffic engineering, which should be investigated and examined. After all, a trade-off should be maintained between cost and efficiency.
3. QoS-based TE in SDN: the diversity of devices in networks and the development of different applications with differing QoS, providing the requested QoS for users encounter solid challenges and problems [128]129]. Hence, new applications such as distance learning and video conferencing must address these different QoS requirements to succeed. Thus, SDN-based traffic engineering should properly manage traffic in core, aggregation, and edge links to provide the requested QoS for the network. As a result, Future studies on sdn-based traffic engineering and QoS issues should be considered and investigated.

1. *Scalability* for better network management and for enhancing the efficiency of network resources, mechanisms and frameworks for discovering elephant flows, flow aggregation, and flow disaggregation are highly needed. Current controllers for big data applications are not appropriate [130]5 because flow table updating requests and transmitting and processing information related to big data are time-consuming. Hence, if SDN controllers manage these applications, the network will encounter some problems. For further research in this area, the scarcity of the off-the-shelf device on the market can induce the limitation in SDN technology that should be considered. Thus, the limit of scalability should be investigated in future studies.
2. *Energy-aware TE mechanisms for SDN* Nowadays, energy consumption is one of the most critical and urgent issues for networks, especially data center networks [17]. SDN features such as network programmability and centralized network view can help introduce new energy-aware traffic engineering mechanisms for networks. Dynamic traffic load balancing and efficient congestion control schemes can improve the energy consumption of networks. So, energy consumption must be considered an important parameter in the following TE mechanisms design. (Table 5)

Table 5 Dependability parameters of each proposed solution

Reference	Proposed Scheme	Solution	Availability	Reliability	Performance
[68]	LBBSRT	Load balancing is used to improve the network reliability & scalability	✔	✔	✔
[97]	HiQoS	A path allocation method with QoS considering & reliability	–	✔	✔
[114]	B4	A layer base structure for bandwidth reallocation	✔	✔	✔
[115]	Microsoft TE	TE for SWAN for reliability and availability provisioning	✔	–	✔
[116]	EFSUTE	efficient and survival traffic engineering for reliability	✔	✔	–
[117]	smart routing system	Smart routing system for reliability	–	✔	–
[118]	Secure-based traffic management	Secure path selection to increase reliability and availability using virtualization	✔	✔	✔

5 Conclusion

SDN has caused a fundamental change and evolution in managing traffic and maintaining network efficiency based on its unique features. The published papers on traffic engineering have tried to fulfill the software defined network goals. Therefore, this paper classifies the papers into topology discovery, traffic measurement, load balancing, QoS, and dependability in SDN. We tried to provide a brief overview of SDN-based architecture in the first section of this paper. Then, we examined different mechanisms and methods proposed for traffic engineering in traditional and SDN-based networks. This review paper outlined the merits and demerits of the existing techniques.

Furthermore, a classification based on the role of traffic engineering methods in SDN networks was given. This categorization of the available traffic engineering methods included the following items: topology discovery, traffic measurement, load balancing, QoS, and dependability. Traffic engineering in SDN networks is at the infant stage of development, with several gaps, challenges, and problems. Interested future researchers are recommended to systematically address and investigate the issues and problems underscored in this overview study.

Funding No funding was received to assist with the preparation of this manuscript.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Hossein Dabbagh, N., & Ghaffari, A. (2017). Protocol for controlling congestion in wireless sensor networks. *Wireless Personal Communications, 95*, 3233–3251.
2. Mohammadi, R., & Ghaffari, A. (2015). Optimizing reliability through network coding in wireless multimedia sensor networks. *Indian Journal of Science and Technology, 8*(9), 834.
3. Wan, J., Zou, C., Zhou, K., Lu, R., & Li, D. (2014). IoT sensing framework with inter-cloud computing capability in vehicular networking. *Electronic Commerce Research, 14*(3), 389–416.
4. Tomovic, S., Yoshigoe, K., Maljevic, I., & Radusinovic, I. (2017). Software-defined fog network architecture for IoT. *Wireless Personal Communications, 92*(1), 181–196.
5. James, M., Chui, M., Brown, B., Bughin, J., Dobbs, R., Ch. Roxburgh, Hung Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity." *McKinsey Global Institute*
6. Goode, B. (2002). Voice over internet protocol (VoIP). *Proceedings of the IEEE, 90*(9), 1495–1517.
7. Szyrkowiec, T. et al., (2014) First field demonstration of cloud datacenter workflow automation employing dynamic optical transport network resources under OpenStack and OpenFlow orchestration. In *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*, pp. 2595–2602. doi: https://doi.org/10.1049/cp.2013.1693.
8. Persico, V., Botta, A., Marchetta, P., Montieri, A., & Pescapé, A. (2017). On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe. *Computer Networks, 112*, 67–83.
9. Karakus, M., & Durresi, A. (2017). Quality of service (QoS) in software defined networking (SDN): a survey. *Journal of Network and Computer Applications, 80*, 200–218. https://doi.org/10.1016/j.jnca.2016.12.019
10. Javadtalab, A., Semsarzadeh, M., Khanchi, A., Shirmohammadi, S., & Yassine, A. (2015). Continuous one-way detection of available bandwidth changes for video streaming over best-effort networks. *IEEE Transactions on Instrumentation and Measurement, 64*(1), 190–203.

11. Choshin, M., & Ghaffari, A. (2017). An investigation of the impact of effective factors on the success of e-commerce in small- and medium-sized companies. *Computers in Human Behavior, 66*, 67–74. https://doi.org/10.1016/j.chb.2016.09.026
12. Nie, L., Jiang, D., Guo, L., & Yu, S. (2016). Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks. *Journal of Network and Computer Applications, 76*, 16–22. https://doi.org/10.1016/j.jnca.2016.10.006
13. Wang, N., Ho, K. H., Pavlou, G., & Howarth, M. (2008). An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys & Tutorials, 10*(1), 36–56.
14. Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., & Xiao, X. (2002). RFC3272: Overview and principles of Internet traffic engineering. *United States*. https://doi.org/10.17487/RFC3272
15. Lu, Y., Ling, Z., Zhu, S., & Tang, L. (2017). SDTCP: Towards Datacenter TCP congestion control with SDN for IoT applications. *Sensors, 17*(1), 109.
16. Yuan, J., Zhao, D., Long, K., & Zheng, Y. (2017). Improved immunization strategy to reduce energy consumption on nodes traffic. *Optics Communications, 389*, 314–317.
17. Xu, G., Dai, B., Huang, B., Yang, J., & Wen, S. (2017). Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. *Future Generation Computer Systems, 68*, 163–174. https://doi.org/10.1016/j.future.2016.08.024
18. Yassine, A., Rahimi, H., & Shirmohammadi, S. (2015). Software defined network traffic measurement: Current trends and challenges. *IEEE Instrumentation & Measurement Magazine, 18*(2), 42–50.
19. Agarwal, S., Kodialam, M., & Lakshman, T. V. (2013). "Traffic engineering in software defined networks. *in INFOCOM Proceedings IEEE, 2013*, 2211–2219.
20. Zhang, X., Guo, L., Hou, W., Zhang, Q., & Wang, S. (2017). Failure recovery solutions using cognitive mechanisms based on software-defined optical network platform. *Optical Engineering, 56*(1), 16107.
21. Xiong, B., Yang, K., Zhao, J., & Li, K. (2016). Efficient and robust dynamic network traffic partitioning based on flow tables. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2016.04.013
22. Shirmarz, A., & Ghaffari, A. (2020). An autonomic software defined network (SDN) architecture with performance improvement considering. *Journal of Information Systems and Telecommunication (JIST), 8*(2), 1–9. https://doi.org/10.29252/jist.8.30.121
23. Shirmarz, A., & Ghaffari, A. (2020). Performance issues and solutions in SDN-based data center: A survey. *The Journal of Supercomputing, 76*, 7545–7593. https://doi.org/10.1007/s11227-020-03180-7
24. Shirmarz, A., & Ghaffari, A. (2021). Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-020-02754-w
25. Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and Computer Applications, 67*, 1–25. https://doi.org/10.1016/j.jnca.2016.03.016
26. Shirmarz, A., & Ghaffari, A. (2022). Network traffic discrimination improvement in software defined network (SDN) with deep autoencoder and ensemble method". *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-022-03810-3
27. McKeown, N., et al. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review, 38*(2), 69–74. https://doi.org/10.1145/1355734.1355746
28. Kumari, A., Member, S., Sairam, A. S., & Member, S. (2019). A survey of controller placement problem in software defined networks. *Networking and Internet Architecture (cs NI)*. https://doi.org/10.1109/ACCESS.2019.2893283
29. Shu, Z., et al. (2016). Traffic engineering in software-defined networking: Measurement and management. *IEEE Access, 4*, 3246–3256.
30. Tahaei, H., Salleh, R., Khan, S., Izard, R., Choo, K.-K.R., & Anuar, N. B. (2017). A multi-objective software defined network traffic measurement. *Measurement, 95*, 317–327. https://doi.org/10.1016/j.measurement.2016.10.026
31. Abbasi, M. R., Guleria, A., & Devi, M. S. (2016). Traffic engineering in software defined networks: A survey. *Journal of Telecommunications and Information Technology, 2016*(4), 3–14.
32. Siripongwutikorn, P., Banerjee, S., Tipper, D., Programs, T.H. (2002) Laboratories, "Traffic Engineering in the Internet : A Survey of Load Balanced," pp. 1–9, *White paper*
33. Abdullah, Z. N., Ahmad, I., & Hussain, I. (2019). Segment routing in software defined networks: A survey. *IEEE Communications Surveys and Tutorials, 21*(1), 464–486. https://doi.org/10.1109/COMST.2018.2869754
34. Bouleanu, I., Bechet, P., & Sârbu, A. (2020). A survey on network planning and traffic engineering for deployable networks. *International conference KNOWLEDGE-BASED ORGANIZATION, 26*(3), 43–48. https://doi.org/10.2478/kbo-2020-0113
35. Priyadarsini, M., Bera, P. (2021). Software defined networking architecture , traffic management , security , and placement : A survey. *Computer Networks*, vol. 192, no
36. Khan, S., Gani, A., Wahab, A. A., Guizani, M., Khan, M. K. (2016) Topology discovery in software defined networks: Threats, Taxonomy, and State-of-the-art. *IEEE Communications Surveys & Tutorials*
37. Saha, A. K., Sambyo, K., Bhunia, C. T. (2016) Topology Discovery, Loop Finding and Alternative Path Solution in POX Controller. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2016, vol. 2.
38. Ochoa-Aday, L., Cervelló-Pastor, C., Fernández-Fernández, A. (2016) A Distributed Algorithm for Topology Discovery in Software-Defined Networks. In *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, Springer, 2016, pp. 363–367.
39. Pakzad, F., Portmann, M., Tan, W. L., & Indulska, J. (2016). Efficient topology discovery in Openflow-based software defined networks. *Computer Communications, 77*, 52–61.
40. Pakzad, F., Portmann, M., Tan, W. L., Indulska, J. (2014) Efficient topology discovery in software defined networks. In *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on*, 2014, pp. 1–8.
41. Scott, C., et al. (2015). Troubleshooting blackbox SDN control software with minimal causal sequences. *ACM SIGCOMM Computer Communication Review, 44*(4), 395–406.
42. Hong, S., Xu, L., Wang, H, Gu, G.(2015) Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. In *NDSS Symposium*
43. Aday, L. O., Pastor, C. C., & Fernández, A. F. (2016). Discovering the network topology: an efficient approach for SDN. *ADCAIJ Advances in Distributed Computing and Artificial Intelligence Journal, 5*, 101–108.
44. Hollander, J. (2007) *A Link Layer Discovery Protocol Fuzzer*. Citeseer
45. Van Adrichem, N. L. M., Doerr, C., Kuipers, F. A. (2014) OpenNetMon : Network Monitoring in OpenFlow Software-Defined Networks. In *OpenNetMon : Network Monitoring in OpenFlow Software-Defined Networks*
46. Malboubi, M., Wang, L., Chuah, C.-N., Sharma, P.(2014). Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp). In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 934–942.

47. Tootoonchian, A., Ghobadi, M., Ganjali, Y. (2010). OpenTM: traffic matrix estimator for OpenFlow networks. In *International Conference on Passive and Active Network Measurement*, pp. 201–210.
48. Chowdhury, S. R., Bari, M. F., Ahmed, R., & Boutaba, R. (2014). "Payless: A low cost network monitoring framework for software defined networks", in. *IEEE Network Operations and Management Symposium (NOMS), 2014*, 1–9.
49. Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., Madhyastha, H. V (2013). "Flowsense: Monitoring network utilization with zero measurement cost. In *International Conference on Passive and Active Network Measurement*, pp. 31–41.
50. Moshref, M., Yu, M., Govindan, R., & Vahdat, A. (2014). DREAM: Dynamic resource allocation for software-defined measurement. *ACM SIGCOMM Computer Communication Review, 44*(4), 419–430.
51. Sun, P., Yu, M., Freedman, M. J., Rexford, J., & Walker, D. (2015). Hone: Joint host-network traffic management in software-defined networks. *Journal of Network and Systems Management, 23*(2), 374–399.
52. Suh, J., Kwon, T. T., Dixon, C., Felter, W., Carter, J.(2014) OpenSample: A low-latency, sampling-based measurement platform for commodity SDN. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pp. 228–237.
53. Yu, M., Jose, L., Miao, R. (2013) Software Defined Traffic Measurement with OpenSketch. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 29–42.
54. Rasley, J., et al. (2015). Planck: Millisecond-scale monitoring and control for commodity networks. *ACM SIGCOMM Computer Communication Review, 44*(4), 407–418.
55. "NetFlow,"https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.
56. Liu, C. Malboubi, Am., Chuah, C.-N.(2016) OpenMeasure: Adaptive flow measurement & inference with online learning in SDN. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pp. 47–52.
57. Hark, R., Stingl, D., Richerzhagen, N., Nahrstedt, K., Steinmetz, R.(2016) DIStTM: Collaborative Traffic Matrix Estimation in Distributed SDN Control Planes. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, 2016.
58. Shirali-Shahreza, S., Ganjali, Y.(2013) FleXam: flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 167–168.
59. Ballard, J. R., Rae, I., Akella, A. (2008) Extensible and Scalable Network Monitoring Using OpenSAFE. In *2010 Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN 10)*
60. Yu, Y., Qian, C., Li, X. (2014). Distributed and collaborative traffic monitoring in software defined networks. In *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 85–90.
61. Phemius, K., Bouet, M., & Leguay, J. (2014). DISCO: Distributed multi-domain SDN controllers. *In IEEE Network Operations and Management Symposium (NOMS), 2014*, 1–4. https://doi.org/10.1109/NOMS.2014.6838330
62. Wang, M.-H., Wu, S.-Y., Yen, L.-H., & Tseng, C.-C. (2016). "PathMon: Path-specific traffic monitoring in OpenFlow-enabled networks", in *Ubiquitous and Future Networks (ICUFN). Eighth International Conference on, 2016*, 775–780.
63. Yang, G., Lee, K., Jeong, W., Yoo, C.(2016) Flo-v: Low overhead network monitoring framework in virtualized software defined networks. In *Proceedings of the 11th International Conference on Future Internet Technologies*, pp. 90–94.
64. Grover, N., Agarwal, N., Kataoka, K.(2015) liteFlow: Lightweight and distributed flow monitoring platform for SDN. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, 2015, pp. 1–9.
65. Su, Z., Hamdi, M. (2015) MDCP: Measurement-Aware Distributed Controller Placement for Software Defined Networks. In *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*, pp. 380–387.
66. Su, Z., Wang, T., Hamd, M.(2015) COSTA: Cross-layer optimization for sketch-based software defined measurement task assignment. In *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, pp. 183–188.
67. Lin, S.-C., Wang, P., & Luo, M. (2016). Control traffic balancing in software defined networks. *Computer Networks, 106*, 260–271. https://doi.org/10.1016/j.comnet.2015.08.004
68. Zhong, H., Fang, Y., & Cui, J. (2017). LBBSRT: An efficient SDN load balancing scheme based on server response time. *Future Generation Computer Systems, 68*, 183–190.
69. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.(2010) Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI*, 2010, vol. 10, p. 19.
70. Curtis, A. R., Kim, W., & Yalagandula, P. (2011). Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", in *INFOCOM. Proceedings IEEE, 2011*, 1629–1637.
71. Benson, T., Anand, A., Akella, A., Zhang, M.(2011) MicroTE: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, p. 8.
72. Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., & Banerjee, S. (2011). DevoFlow: Scaling flow management for high-performance networks. *ACM SIGCOMM Computer Communication Review, 41*(4), 254–265.
73. Trestian, R., Muntean, G.-M., Katrinis K.(2013) MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 904–907.
74. Yu, M., Rexford, J., Freedman, M. J., & Wang, J. (2010). Scalable flow-based networking with DIFANE. *ACM SIGCOMM Computer Communication Review, 40*(4), 351–362.
75. Cimorelli, F., Priscoli, F. D., Pietrabissa, A., Celsi, L. R., Suraci, V., Zuccaro, L.(2016). A distributed load balancing algorithm for the control plane in software defined networking. In *Control and Automation (MED), 2016 24th Mediterranean Conference on*, 2016, pp. 1033–1040.
76. Liu, G., Trotter, M., Ren, Y., Wood, T.(2016) NetAlytics: Cloud-Scale Application Performance Monitoring with SDN and NFV In *Proceedings of the 17th International Middleware Conference*, p. 8.
77. Guo, Y., Wang, Z., Yin, X., Shi, X., Wu, J.(2014) Traffic engineering in SDN/OSPF hybrid network. In *2014 IEEE 22nd International Conference on Network Protocols*, pp. 563–568.
78. Dinh, K. T., Kukliński, S., Kujawa, W., Ulaski, M.(2016) MSDN-TE: Multipath Based Traffic Engineering for SDN. In *Asian Conference on Intelligent Information and Database Systems*, pp. 630–639.
79. Paquin, F., Rivnay, J., Salleo, A., Stingelin, N., Silva, C.(2015) A Deep Learning Based System for Traffic Engineering in Software Defined Network. *J. Mater. Chem. C*, vol. 3, no. Dl, pp. 10715–10722, 2015, doi: https://doi.org/10.1039/b000000x.
80. Deepshikha, M. Dave (2019) An efficient traffic management solution in data center networking using SDN. In *2018 International Conference on Power Energy, Environment and Intelligent Control, PEEIC 2018*, pp. 825–829. doi: https://doi.org/10.1109/PEEIC.2018.8665589.

81. Mohammadi, R., Javidan, R., Keshtgari, M., & Akbari, R. (2018). A novel multicast traffic engineering technique in SDN using TLBO algorithm. *Telecommunication Systems, 68*(3), 583–592. https://doi.org/10.1007/s11235-017-0409-x
82. Cheng Ren, Y. L., Shiwei, B., Yu W. (2020) Achieving near-optimal traffic engineering using a distributed algorithm in hybrid SDN. *IEEE Access*, vol. 8
83. Bastam, M., RahimiZadeh, K., & Yousefpour, R. (2021). Design and performance evaluation of a new traffic engineering technique for software-defined network datacenters. *Journal of Network and Systems Management, 29*(4), 38. https://doi.org/10.1007/s10922-021-09605-9
84. 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*, pp. 1–7.
85. Koponen, T. et al. (2010) Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, pp. 351–364. doi: 10.1.1.186.3537.
86. Hu, Y., Wang, W., Gong, X., Que, X., Cheng, S. (2012) Balanceflow: controller load balancing for openflow networks. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 2, pp. 780–785.
87. 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*, 2012, pp. 19–24.
88. Gao, X., Kong, L., Li, W., Liang, W., Chen, Y., Chen, G.(2016) Traffic load balancing schemes for devolved controllers in mega data centers. *IEEE Transactions on Parallel and Distributed Systems*
89. Teng, Y., Xia, Z.(2020) A traffic engineering technology based on segment routing in SDN. In *Proceedings - 16th International Conference on Mobility, Sensing and Networking, MSN*, pp. 636–641. doi: https://doi.org/10.1109/MSN50589.2020.00106.
90. Shirmarz, A., & Ghaffari, A. (2019). An adaptive greedy flow routing algorithm for performance improvement in a software-defined network. *International numerical modeling: Electronic networks, Devices, and Fields-Wiley online library, 33*(1), 1–21. https://doi.org/10.1002/jnm.2676
91. Alireza Shirmarz, A. G. (2021). (2021) automatic software defined network (SDN) performance management using TOPSIS Decision-Making Algorithm. *Journal of Grid Computing, 19*, 1–21. https://doi.org/10.1007/s10723-021-09557-z
92. Tajedin, F., Farhoudi, M., Samiei, A., Akbari, B.(2019) DTE:Dynamic Traffic Engineering in Software Defined Data Center Networks. In *2019 International Conference on Computer Engineering, Network, and Intelligent Multimedia, CENIM 2019 - Proceeding*, 2019, vol. 2019-Novem. https://doi.org/10.1109/CENIM48368.2019.8973350.
93. Bera, S., Misra, S., Saha, N.(2018) DynamiTE: Dynamic traffic engineering in software-defined cyber physical systems. *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018 - Proceedings*, pp. 1–6, 2018, doi: https://doi.org/10.1109/ICCW.2018.8403550.
94. Shirmarz, A., Ghaffari, A.(2021) A novel flow routing algorithm based on non-dominated ranking and crowd distance sorting to improve the performance in SDN. *Photonic Network Communications*, no. 0123456789, 2021, doi: https://doi.org/10.1007/s11107-021-00951-x.
95. Dinh, K. T., Kukliński, S., Osiński, T., & Wytrębowicz, J. (2020). Heuristic traffic engineering for SDN. *Journal of Information and Telecommunication, 4*(3), 251–266. https://doi.org/10.1080/24751839.2020.1755528
96. Win. M.T.Z. (2019) Application-aware Traffic Engineering in Software Defined Networking*" (Doctoral dissertation, MERAL Portal)*
97. Yan, J., Zhang, H., Shuai, Q., Liu, B., & Guo, X. (2015). HiQoS: An SDN-based multipath QoS solution. *China Communications, 12*(5), 123–133. https://doi.org/10.1109/CC.2015.7112035
98. Egilmez, H. E., Dane, S. T.(2012) OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2012, pp. 1–8.
99. Ongaro, F., Cerqueira, E., Foschini, L., Corradi, A., & Gerla, M. (2015). "Enhancing the quality level support for real-time multimedia applications in software-defined networks. *In Computing, Networking and Communications (ICNC) International Conference on, 2015*, 505–509.
100. C. Thorpe, A. Hava, J. Langlois, A. Dumas, and C. Olariu, "iMOS: Enabling VoIP QoS Monitoring at Intermediate Nodes in an OpenFlow SDN,", *In IEEE International Conference on Cloud Engineering Workshop (IC2EW) (pp. 76–81). IEEE, 2016*. doi: https://doi.org/10.1109/IC2EW.2016.33.
101. Alidadi, A., Arab, S., & Askari, T. (2021). A novel optimized routing algorithm for QoS traffic engineering in SDN-based mobile networks. *ICT Express, 8*(1), 130–134. https://doi.org/10.1016/j.icte.2021.12.010
102. Huang, X., Zeng, M., & Xie, K. (2021). (2021) Intelligent traffic control for QoS optimization in hybrid SDNs. *Computer Networks, 189*, 107877. https://doi.org/10.1016/j.comnet.2021.107877
103. Matera, F., & Tego, E. (2021). Machine learning for QoE and QoS control of slices in a wide area network test bed. *In AEIT International Annual Conference (AEIT), 2021*, 1–6. https://doi.org/10.23919/aeit53387.2021.9626968
104. Zhang, Y., Gorlatch, S. (2021) Optimizing energy efficiency of qos-based routing in software-defined networks. In *the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2021, pp. 87–94. doi: https://doi.org/10.1145/3479242.3487325.
105. Alkhafaji, A. R., Al-Turaihi, F. S.(2021) Multi-Layer network slicing and resource allocation scheme for traffic-aware QoS ensured SDN/NFV-5G Network. In *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, 2021, pp. 327–331. doi: https://doi.org/10.1109/bicits51482.2021.9509901.
106. Tomovic, S., & Radusinovic, I. (2019). Towards a scalable, robust and QoS-aware virtual-link provisioning in SDN-based ISP networks. *IEEE Transactions on Network and Service Management, 16*(3), 1032–1045. https://doi.org/10.1109/TNSM.2019.2929161
107. Win, M. T. Z., Ishibashi, Y., Mya, K. T.(2019) QoS-aware traffic engineering in software defined networks. *Proceedings of 2019 25th Asia-Pacific Conference on Communications, APCC 2019*, pp. 171–176, 2019, doi: https://doi.org/10.1109/APCC47188.2019.9026524.
108. Lin, C., Bi, Y., Zhao, H., Liu, Z., Jia, S., & Zhu, J. (2018). DTE-SDN: A dynamic traffic engineering engine for delay-sensitive transfer. *IEEE Internet of Things Journal, 5*(6), 5240–5253. https://doi.org/10.1109/JIOT.2018.2872439
109. Thazin, N., Nwe, K. M., Ishibashi, Y.(2019) End-to-end dynamic bandwidth resource allocation based on QoS demand in SDN. *Proceedings of 2019 25th Asia-PacificConference on Communications, APCC 2019*, pp. 244–249, 2019, doi: https://doi.org/10.1109/APCC47188.2019.9026511.
110. B, V. A., Beshley, M., Dutko, L. (2022) Intelligent Traffic Engineering for Future. *Future Intent-Based Networking*, pp. 161–181, doi: https://doi.org/10.1007/978-3-030-92435-5.

111. Sun, P., Guo, Z., Lan, J., Li, J., Hu, Y., & Baker, T. (2021). ScaleDRL : A scalable deep reinforcement learning approach for traffic engineering in sdn with pinning control. *Computer Networks, 190*, 107891. https://doi.org/10.1016/j.comnet.2021.107891
112. Mostafaei, H., Shojafar, M., Member, S., Conti, M., & Member, S. (2021). TEL : Low-latency failover traffic engineering in data plane. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, 18*(4), 4697–4710.
113. Gales, E. M., Croitoru, V.(2020) Traffic engineering and QoS in a Proposed MPLS-VPN. In *2020 14th International Symposium on Electronics and Telecommunications, ISETC 2020 - Conference Proceedings*, pp. 3–6. doi: https://doi.org/10.1109/ISETC50328.2020.9301135.
114. Jain, S. et al.(2013) B4: Experience with a Globally-Deployed Software DefinedWA. In *Proceedings of the ACM SIGCOMM conference on SIGCOMM - SIGCOMM '13*, 2013, pp. 3–14. doi: https://doi.org/10.1145/2486001.2486019.
115. Hong, C.-Y. et al.(2013) Achieving high utilization with software-driven WAN. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, vol. 43, no. 4, pp. 15–26. doi: https://doi.org/10.1145/2486001.2486012.
116. Mohammadi, R., & Javidan, R. (2021). EFSUTE: A novel efficient and survivable traffic engineering for software defined networks. *Journal of Reliable Intelligent Environments*. https://doi.org/10.1007/s40860-021-00139-0
117. Lemeshko, O., Yeremenko, O., Yevdokymenko, M., Zhuravlova, A., Kruhlova, A., Lemeshko, V. (2021) Research of improved traffic engineering fault-tolerant routing mechanism in SD-WAN. *2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering, UKRCON 2021 - Proceedings*, pp. 187–190, 2021, doi: https://doi.org/10.1109/UKRCON53503.2021.9575272.
118. Lemeshko, O., Yeremenko, O., Yevdokymenko, M., Shapovalova, A., Radivilova, T., Ageyev, D. (2002) Secure based traffic engineering model in softwarized networks. In *ATIT 2020 - Proceedings: 2020 2nd IEEE International Conference on Advanced Trends in Information Theory*, 2020, pp. 143–147. doi: https://doi.org/10.1109/ATIT50783.2020.9349301.
119. Soua, R. et al.(2017) SDN coordination for CCN and FC content dissemination in VANETs. In *Ad Hoc Networks*, Springer, 2017, pp. 221–233.
120. Blanco, B. et al.(2017) Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. *Computer Standards & Interfaces*
121. Galluccio, L., Milardo, S., Morabito, G., & Palazzo, S. (2015). "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. *In IEEE Conference on Computer Communications (INFOCOM), 2015*, 513–521.
122. Luo, T., Tan, H.-P., & Quek, T. Q. S. (2012). Sensor OpenFlow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters, 16*(11), 1896–1899.
123. Di Dio, P., et al. (2016). Exploiting state information to support QoS in software-defined WSNs. *In Ad Hoc Networking Workshop (Med-Hoc-Net) Mediterranean, 2016*, 1–7.
124. Fotouhi, H., Vahabi, M., Ray, A., Björkman, M.(2016) SDN-TAP: An SDN-based traffic aware protocol for wireless sensor networks. In *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*, 2016, pp. 1–6.
125. Rong, B., Qiu, X., Kadoch, M., Sun, S., Li, W.(2016) Intelligent SDN and NFV for 5G HetNet Dynamics. In *5G Heterogeneous Networks*, Springer, pp. 15–40.
126. Zhang, D., Guo, H., Yang, T., Wu, J.(2017) Optical Switching based Small-world Data Center Network. *Computer Communications*
127. Benamrane, F., Ros, F. J., Ben Mamoun, M. (2016) Synchronisation cost of multi-controller deployments in software-defined networks. *International Journal of High Performance Computing and Networking*, vol. 9, no. 4, pp. 291–298.
128. Rangisetti, A. K., T. V. Pasca S., Tamma, B. R. (2017) QoS Aware load balance in software defined LTE networks. *Computer Communications*, vol. 97, pp. 52–71, doi: https://doi.org/10.1016/j.comcom.2016.09.005.
129. Zuo, Y., Wu, Y., Min, G., & Cui, L. (2019). Learning-based network path planning for traffic engineering. *Future Generation Computer Systems, 92*, 59–67. https://doi.org/10.1016/j.future.2018.09.043
130. Gahlot, L. K., Khurana, P., Hasija, Y. (2017) A Smart Vision with the 5G Era and Big Data—Next Edge in Connecting World. In *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, Springer, 2017, pp. 151–170.

Ramin Mohammadi received a Computer Engineering degree from Teraktorsazi Tabriz University, Iran, in 2009 and a Master of Computer Engineering from the Islamic Azad University of Shabestar in 2014. He is currently a Ph.D. student in computer engineering at OMU University, Samsun, Turkey. His research Activities include wireless sensor networks, automatic learning devices and combining these two issues.

Sedat Akleylek received a B.Sc. degree in Mathematics majoring in Computer Science, from Ege University in 2004 in Izmir, Turkey. His M.Sc. and Ph.D. in Cryptography are from Middle East Technical University in 2008 and 2010, in Ankara, Turkey, respectively. He is currently employed as an associate professor at the Department of Computer Engineering, Ondokuz Mayis University, Samsun, Turkey, since 2016. He is also affiliated with the Cryptography Program at IAM, METU, Ankara, Turkey. He had a postdoctoral researcher position at the Cryptography and Computer Algebra Group, TU Darmstadt from July 2014 to July 2015. His research interests include in the areas of post-quantum cryptography, cryptographic functions, algorithms and complexity, architectures for computations in finite fields and machine learning for cyber security.

Ali Ghaffari received his BSc, MSc and Ph.D. degrees in computer engineering from the University of Tehran and IAU (Islamic Azad University), TEHRAN, IRAN in 1994, 2002 and 2011 respectively. He has served as a reviewer for some high-ranked journal such as Applied Soft Computing, Ad Hoc networks, Future Generation Computer System (FGCS), Journal of Ambient Intelligent and Humanized Computing (AIHC) and computer networks. Dr. Ghaffari has been featured among the World's Top 2% Scientists List in computer science, according to a conducted study by US-based Stanford University in 2020 and 2021. As an associate professor of computer engineering at Islamic Azad University, Tabriz branch, IRAN, his research interests are mainly in the field of software defined network (SDN), Wireless Sensor Networks (WSNs), Mobile Ad Hoc Networks (MANETs), Vehicular Ad Hoc Networks (VANETs), networks security and Quality of Service (QoS). He has published more than 100 international conference and reviewed journal papers.

Alireza Shirmarz received a Diploma in Mathematics and Physics from Falsafi School, a bachelor's degree in Computer engineering from Tehran Shahed University, Iran, and the Master's Science degree in network engineering from Amirkabir University of Technology (Tehran Polytechnic), Iran. He succeeded in receiving his Ph.D. from the Tehran north branch of IAU. He is currently a researcher at Amirkabir University (Tehran Polytechnic) and is interested in SDN, NFV, network traffic Analysis, and optimization. He has also been teaching at the Aletaha Institute of Higher Education as an Assistant Professor.