Wireless Personal Communications (2021) 116:2593–2614
https://doi.org/10.1007/s11277-020-07812-2

A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)

Surendra Kumar Keshari[1,2] · **Vineet Kansal**[3] · **Sumit Kumar**[4]

Published online: 24 September 2020

Abstract
In modern life the internet has become backbone of digital society which is the packet switched distributed network and connected with almost every digital devices, available everywhere in the world. The traditional network carries few challenges like, dependency on vendors, difficulty to manage large network, dynamically changing of forwarding policies and more. To overcome such challenges, today the idea of Software Defined Networking (SDN) came into existence. The basic idea behind SDN is to implement programmable network by separately managing the control plane and data plane to improve the efficiency of network performance. The main problem with SDN is the Controller Placement Problem (CPP), which gives the overview of whole network. Today the main focus of the researchers is to solve the CPP. CPP is a NP-hard problem because the network should consist of minimum controllers and controllers should be placed on appropriate locations. For the large size network, controller deployment is difficult to manage. But the challenge in this area is Quality of Services (QoS) in respect of controller management. This paper investigates the systematic review of QoS based on controller's problems, analyzed the current research and summarized the findings of the different controller's performance based on QoS parameters e.g. reliability, scalability, consistency and load balancing. Finally, this paper also highlights the research challenges to improve the QoS in SDN.

Keywords SDN · QoS · Controller · Scalability · Reliability · Consistency · Load balancing

✉ Surendra Kumar Keshari
surendra.keshari@gmail.com

[1] Department of Information Technology, KIET Group of Institutions, Delhi-NCR, Ghaziabad, India

[2] Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India

[3] Department of Computer Science and Engineering, Institute of Engineering and Technology, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India

[4] Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India

1 Introduction

Computer network is a large network of computers connected with different interconnecting nodes like; repeaters, switches, routers and many more intermediate nodes. The modern computer network is a distributed, packet switched network which is connected with almost every digital device and available everywhere in the world. The traditional network carries few challenges like, dependency on vendors, difficulty to manage large network, dynamically changing of forwarding policies and more. To overcome the challenges of existing conventional network, the concept of programmable network came into existence [1]. The behavior of programmable network is governed by software program which is called software defined networks-(SDN). This programmable network is a dynamic emerging network-infrastructure which separately manages data plane and control plane, in contrast to conventional networks [1]. The control plane is logically separated and centralized where as data forwarding data plane follows the decision of control plane [2]. The architecture of SDN is shown in Fig. 1 and explained below.

a. **Application Layer (Management Plane):** This layer includes different networking services and applications that manages network behaviors like, load balancing, firewalls etc. The management plane configures and monitors the network devices.
b. **Control Layer (Control-Plane):** This control layer behaves like an instructor and is a brain of software programmable network. The control layer takes all the forwarding decisions based on topology and also manages different controllers and this plane.
c. **Infrastructure Layer (Data Plane):** The plane which actually forward the packets based on data-forwarding tables. The data-forwarding layer is physical device which receives instructions from control plane and forward accordingly.
d. **Northbound Interface:** This interface is basically a communication mediator between upper and middle plane, management and control. The instructions of application plane are passed to controllers though this interface.

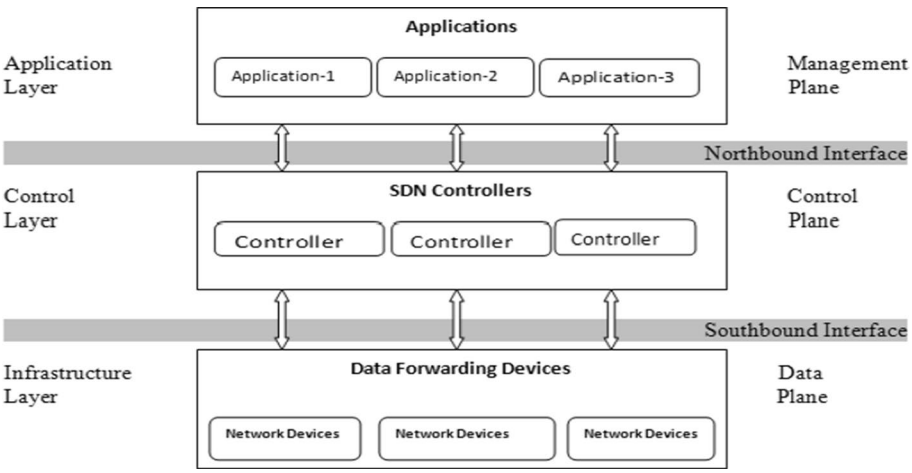

Fig. 1 Software-defined-networks architecture

e. **Southbound Interface:** This interface is basically a communication mediator between control and infrastructure layer. The OpenFlow protocol is used to decouple these two layers.

The OpenFlow protocol is basically a communication interface between OpenFlow controllers and OpenFlow forwarding planes [3]. This is first standard communication protocol for SDN environments. OpenFlow provides the flow route path to forward the packets across the programmable networks. The key benefit of OpenFlow protocol that it permits different vendor's switches can be configured with controllers. Different versions of OpenFlow are available in SDN environments. This is the management of OpenFlow protocol that the controller gives the instructions to switches that in what manner they handle the incoming data packets. The OpenFlow protocol manages two switch components: The Flow table and The Secure communication channel which includes various details like, encrypted channel, traffic monitoring, managing incoming packets generated by different controllers and many more [4]. The Comparison of SDN Networks and Conventional networks are given in Table 1.

This paper is critical review that focuses on SDN challenges and controller placement problems (CPP) to enhance the QoS of SDN network. The SDN controllers have complete overview of networks and to maintain the QoS in networks controllers must be positioned properly. The challenges that can be affect the QoS due to controllers may be network security, management of network, effectively use of resources, network programmability, managing of network services etc. Such challenges of SDN can be reduced by maintaining the reliability, scalability, consistency and load balancing issues [5–8].

SDN is an attractive research area in today's computer networks communication. But the challenge in this area is QoS in respect of controller management. For the large size network controller deployment is difficult to manage. The reliability and scalability are key challenges for QoS in programmable network. Thus the researchers are doing research to overcome these challenges to answer some questions like, how much controllers required in the SDN infrastructure, find the location to deploy these controllers and also the communication of these controllers with the attached devices [2]. The main objectives of this paper are discussed as per taxonomy and also analyzed the controller placement metrics in respect of QoS parameters.

The main contributions of this paper include the summary of available controllers, their architectural issues and QoS in open source controllers. The paper also covers the analysis of different controller's performance based on QoS parameters and finally, highlighted the research challenges that may improve the QoS of SDN.

The remaining paper is arranged as followings; the Sect. 2 provided the summary of exiting controller's platforms and discussed the various architectural issues of SDN network. Section 3 discussed the QoS in open source controllers. The Sect. 4 analyzed and tabulated the different controller's performance based on certain QoS parameters. In Sect. 5, we have highlighted the research challenges and finally in Sect. 6, we have concluded this paper.

Table 1 Comparison of SDN Networks and Conventional networks

Sl. no.	Conventional networking	Software defined-networking
1	A static network which possess flexibility and agility.	This is a programmable network which can be flexible as per requirements
2	Lower layer and middle layer are coupled in a single hardware device.	Lower layer and middle layer are decoupled and communicate through OpenFlow protocol
3	There are distributed control plane with the hardware switch.	There are logically centralized control plane in SDN environment.
4	Due to manual configuration of switches the network is error prone	Here the network is centralized managed are configured automatically as per need.
5	The maintenance cost is higher	Here maintenance cost less as compare to traditional network
6	There is no concept of centralized network view	Here controller has centralized network view
7	Here network management is very difficult	Network management is easy due to deployment of controllers
8	Scaling of network is not possible in effective manner	Due to decouple Lower layer and middle layer, scaling of network is easy
9	The network management is costly	The network management cost is less
10	Consist the hardware network devices	Generally configured over open source software

2 Controllers Platforms and Architectural Issues

The taxonomy of controller placements on QoS parameters are summarized in Fig. 2. The taxonomy includes the summary of exiting controller's platforms as well as discussed the various architectural issues and given the QoS in open source controllers. Here also analyzed and tabulated the different controller's performance based on certain QoS parameters.

2.1 Existing Controllers' Platforms

There are many Open sources as well as commercial SDN controllers are available. The different features of various controllers' platforms are suited to different applications. Basically the controllers are categorized as distributed controller and centralized controller. The centralized controller implements the control plane logic from a single location and generally suffers scalability issues due to limited capacity of controller. But the distributed controller has no scalability issues and having benefit of high performance when traffic load is high. The brief comparisons of different controllers in respect of different features are listed in Table 2.

Beacon is an OpenFlow Controllers which handles multithreading operations and also event based operations. Beacon is capable for high computing and can handles around greater than twelve million per second. The problem with beacon is its security issues because any change in data compromises the reliability of data [7, 9]. Maestro is a Java Based which provide interface to implement SDN network to manage different applications. This is a high performance controller to manage the switches. Such controllers are

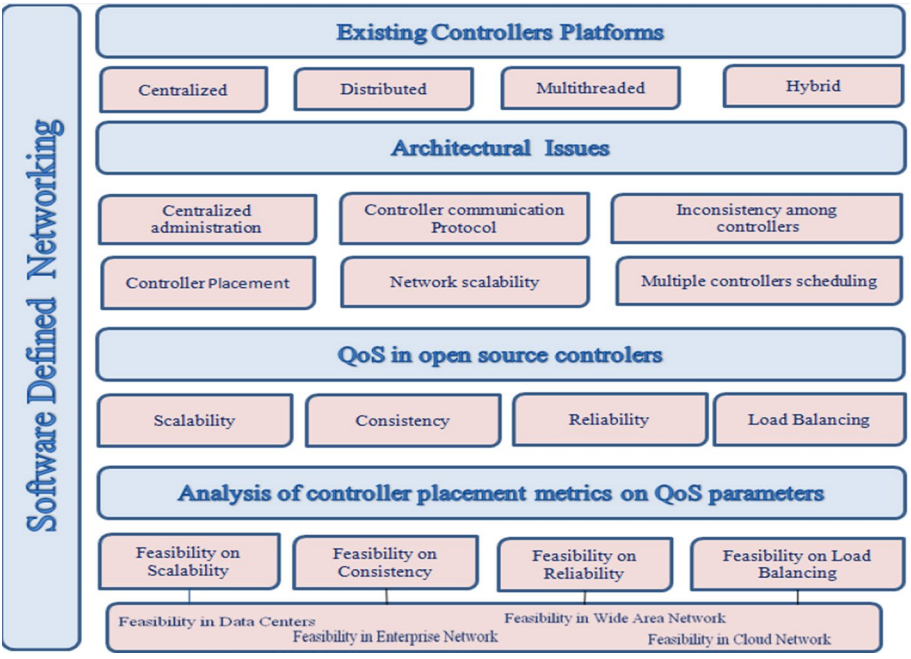

Fig. 2 Taxonomy of controller placements on QoS parameters

capable for optimization techniques to give higher performance output. Maestro is capable to handle the about Ten million of flows per second. This allows the batches of flows by using threaded mechanism [1, 11]. Open MUL, OpenFlow Controllers which handles multithreading operations in SDN and also support multilevel northbound interfaces for attached application. Open MUL supports C Language and used for northbound API. It is able to enhance scalability and performance of SDN networks [7, 13].

RYU is SDN controllers which logically manages APIs in programmable network at upper layer for different applications. Clouds and data centers environments are targeted by RYU. RYU supports in RYU ver1.0, RYU ver1.2 and RYU ver1.3 to manage the traffic. This is developed in python by NTT cloud data centers [7, 14]. Floodlight controllers are based on JAVA which is implementations of Beacon and supports virtual as well as physical switches. These controllers can handles up to 1.39 Millions requests per second on the scale of around 12 threads. This controller is basically designed for large data centers and high performance machines to achieve throughput. Floodlight also defined their own northbound APIs. Like some other controllers Floodlight also compromises security issues because any change in data compromises the reliability of data [7, 21].

POX is the first OpenFlow controller, which drawback is least response time. This controller has security challenges issues because any change in data compromises the reliability of data. This is a Python supported controller. OpenFlow can be easily run on POX platforms [1, 15].

OpenDaylight Java based that support OSGi Frameworks for controllers within autonomous system and also support REST in North bound's API. These controllers are available in both CLI and GUI mode. OpenDaylight is based on a framework where different models like, RESTCONF and NETCONF with their services and functions. OpenDaylight provides a configuration platform to maintain scalability and performance in SDN networks. It supports northbound APIs like, REST and JAVA APIs. OpenDaylight separates the Northbound and Southbound APIs with a SAL (Service Adaption Layer). It supports southbound APIs like: OpenFlow, OVSDB,PCEP, BGP, SNMP. Programmable Network Controller Implements in Data Centers and supports OpenDaylight [1, 17]. NOX, the first OpenFlow controller which was initially single threaded controller but now it supports multithreaded environment i.e. NOX-MT. It can handle fast input and number of requests efficiently. It forwards the packet by mapping of MAC and port numbers from switch hash table. These hash table supports in read operations and new MAC addresses are updated in same hash table [1, 19].

Hyperflow is a distributed controller which is logically centralized. It reduces the overheads by activating communications with other distributed controllers. The Hyperflow implemented through NOX design. In this design each controller's program is directly control individual switch and controllers send a message periodically to show their existence in network to control the switches [7, 10].

ONOS (Open Network Operating Systems) SDN controllers supports network scalability in WAN and available Open source. This is distributed controllers which are implemented to with the aim to manage Network's performances, scalabilities and availabilities issues. ONOS is capable to handle the about one million of flows per second. This is centralized controller which has the capability to monitor the network globally and enhance the network's performance [7, 12]. OpenContrail is an Open Source Controller, which combines the protocols like: BGP, SNMP and some others with SDN technology. These controllers are available in both CLI and GUI mode both which supports northbound API [7, 16]. Ericsson SDN controllers are based on OpenDaylight. and binds the Policy Control that drives end users services personalization within networks [7, 18]. HP VAN SDN

Table 2 Summary of different controllers

Sl no.	Controller name	Organization	Programming language	Open source	Flows/second	Modularity	Productivity	Consistency	Fault	Architecture	References
1	Beacon	Stanford University	Java	Yes	12.8 M	Fair	Fair	No	No	Centralized Multi-threaded	[7, 9]
2	Hyperflow	University of Toronto	C++		30 K	Fair	Fair	Weak	Yes	Distributed	[7, 10]
3	Maestro	Rice Univ.	Java	Yes	4.8 M	Fair	Fair	No	No	Centralized Multi-threaded	[1, 11]
4	ONoS	ON.Lab	Java	Yes	1 M	High	Fair	Weak, strong	Yes	Distributed	[7, 12]
5	OpenMUL	KulCloud	C	Yes				No	No	Centralized Multi-threaded	[7, 13]
6	RYU	NTT	Python	Yes		Fair	Fair	No	No	Centralized Multi-threaded	[7, 14]
7	POX	Nicira	Python	Yes	1.8 M	Low	Fair	No	No	Centralized	[1], [15]
8	OpenContrail	Juniper	Python	Yes		Fair	Fair	No	No	Distributed	[7, 16]
9	OpenDaylight	Linux Foundation	Java	Yes	106 K	High	Fair	Weak	No	Centralized	[1, 17]
10	Ericsson SDN Controller	Ericsson	Java	No						Distributed	[7], [18]
11	NOX	Nicira	Python	Yes	1.8 M	Low	Fair	No	No	Centralized	[1], [19]
12	HP VAN SDN Controller	HP	Java	No						Distributed	[7, 20]
13	FloodLight	Big Switch Network	Java	Yes		Fair	Fair	No	No	Centralized multi-threaded	[7, 21]
14	Onix	Nicira Networks	C, Python	Yes	2.2 M	Fair	Fair	strong	Yes	Distributed	[7, 22]

controllers are based on OpenDaylight which supports OpenFlows protocols and cluster implementation for High-Available, multi applications that includes persistence feature, integrations of OpenStacks and function Chaining. This is an Open Source Controller, which combines the protocols like: BGP, SNMP and some others with SDN technology. These controllers supports number of southbound APIs like: OpenFlow, L3 Agent, L2 Agent [7, 20]. Onix is a distributed controller which is logically centralized. It decreases the overhead by enabling communication with local controllers. This controller facilitates traffic engineering, routing and network scalability. This is Multilanguage supported controller e.g. C++, JAVA, Python based controller which handles the system failure situations. Onix controllers manage faults tolerance and supports data consistency models. Onix supports southbound APIs e.g. OVSDB and OpenFlow [7, 22].

Some controllers are centralized and multithreaded. These controllers are interactively diversed with the different API in northbound interface. Like MUL, Onix, SDN unified controller, floodlight and Meridian are diversified northbound controllers [7]. The SDN network is critically dependent on theses different controller's platforms. The interoperability issues are resolved in southbound API through OpenFlow protocols. Open SDN Controller Based on OpenDaylight and supports MPLS protocols for robust, embedded applications of Cisco. Huawei IP SDN Controller Based on OpenDaylight that manage network controls for Huawei's multilayer APIs.

2.2 Architectural Issues

In SDN environment switches are decoupled with controllers and dump switches. For a large size network, all the controller and switches are active in networks. To enhance the performance in network the controllers' designs as well as the placements of different controllers are still research issues. In various aspects to enhance the network performance other issues like flexibility, scalability, latency, security and consistency are also important [3, 8, 23].

2.2.1 Centralized Administration

To balance traffic load in large scale networks, multiples controller are deployed in SDN networks. When controllers increased in the networks, then centralized management concept varies. To achieve the good QoS of network, multiple controllers are required. Because the multiple controllers' capacity are greater than a single one [23]. The different controller may have different features, so their centralized administration is challenging [24].

2.2.2 Network Scalability

Due to the multiple controller software defined network is easy to scale. But scalability may be challenged QoS due to load balancing among controllers [23, 24].

2.2.3 Inconsistency Among Controllers

Due to multiple controller implementations in programmable network, the major problem is to synchronize the network state information in SDN with multiple controllers. This problem is known as consensus problem [23]. Due to the complex implementation and

incremental latency the consensus approaches are not well suitable. To design the multiple controllers, the consistency among controllers is essential.

2.2.4 Controller Placement

Using only one controller in Software-Defined-Network has many advantages like: managing, controlling, monitoring entire network environment by a single node centrally. But same time inherits the problems of reliability and scalability in the network [23]. As the network grows these problems become worse the performance. The problem of controller placement is known since 2012. To shade the overhead delay and to enhance overall performance of network, there is need to be placed optimum number of controllers at proper distance into networks [2]. In large scaled networks, the deployment of controllers considers two essential questions: (1) what are the numbers of controller required in the networks (2) on which location these controller are installed in networks? These are NP-hard problems, but essential to deploy the multiple controllers [2, 23].

2.2.5 Controller Communication Protocol

In a distributed environment the numbers of controllers are required and they directly affect the QoS of SDN network. To overcome this problem the different controller communication required in effective manner. To communicate controllers there are east–west interface is required. But, in SDN network, the standard protocol needs to be devised in east–west communication. Currently a global network supports the Border Gateway Protocol (BGP) for east–west interfaces [8, 23].

2.2.6 Multiple Controllers Scheduling

The QoS can be better due to multiple controllers but the challenge is here to schedule the different controllers to avoid the overload on any one controller [23]. Here the major challenge is how to balance the overload quickly.

3 QoS in Open Source Controllers

There are many Open sources as well as commercial SDN controllers are available. The different features of various controllers' platforms are suited to different applications. Basically the controllers are categorized as distributed controller and centralized controller. The number of controllers has directly impacts in QoS performance of SDN networks. The controller placement problem may impacts some other problems e.g. controller to controller communication delay (east–west interface communication delay), controller to switch communication.

Floodlight controllers are Java based supported in both OpenFlow virtual and physical switch. These controllers are implementation of Beacon controllers. The Floodlight controller implements the QoS module which provides the features like, flow deletion, flow insertion and some policies to handle the QoS. These modules are implemented in OpenFlow version 1.0 protocol which is able to tracking the policies in switches and applies the policies for services class. In the Floodlight Northbound interface the module QueuePusher

generates the messages for queue configurations for Creating, Reading, Updating and Deleting functions to manage the Open vSwitches [5].

The open source java based OpenDaylight (ODL) controller implements the SDN infrastructure. The OpenDaylight supports OSGi frameworks programmability for controllers located locally and REST frameworks programmability for controllers located remotely. The ODL platform is developing some applications (e.g. Virtualization coordinators and Distributed DoS protection) and few southbound protocols plugins (e.g. NetCONF, BGP, OpenFlow and SNMP) for the heterogeneous network. To get QoS in flow based network the ODL-Lithium developed a southbound plugin for the DOCSIS infrastructure. Similarly another southbound plugin OVSDB developed for ODL-Lithium that configure and manages queues in SDN switches. And also for the QoS an application the Packet Cable Multimedia (PCMM) provides an interface which manages the flow services for the CMTS networks [5]. Here is the another addition in ODL reservation module that aims is to provide the low level resource reservation which gives the network connectivity services, bandwidth, ports to the users for specific allocated time.

The ONOS controllers are Java based supported in both OpenFlow virtual and physical switch. These controllers are implementation of Beacon controllers. This provides the distributed platforms which improves performance, scalability and available networks to service providers. But an ONOS support OpenFlow mechanism due to this method the existing available switches rarely implements the ONOS. So an ONOS platform provides limited QoS because this implementation supports OpenFlow set_queue functionalities. To improve the QoS a SetQueueInstruncton (high level instruction) is implemented in ONOS libraries [5]. The QoS parameters reliability, scalability, consistency and load balancing are briefed below.

3.1 Scalability

The controller's performance in SDN is more concerned. As network increases, it's very difficult by a single controller to manage entire network. To maintain the network efficiency in distributed SDN network, multiple controllers are deployed and networks are scaled. The main problem with SDN is the placement of controller which gives the overview of whole network. Today the main focus of the researchers is to solve the controller placement problem (CPP). CPP is a NP-hard problem because the network should consist of minimum controllers and controllers should be placed on appropriate locations [2]. For the large size network, controller deployment is difficult to manage. But the challenge in this area is Quality of Services (QoS) in respect of controller management.

3.2 Consistency

In distributed controllers due to architectural issues controller may impose inconsistence flow rules in switches and may result instability in programmable networks. The instability in network may cause service interruption, packet loss and asynchronous message communications among controllers [25]. DevoFlow and DIFANE improve such issues by implementing a partial control metric in switches which is contradict to SDN design principals. To maintain consistency few researchers discussed some solutions like, adaptive consistency model, tunable consistency model, clustering techniques, self- adaptive consistency

models and more [11]. However the issues of consistency among controllers cannot be a guaranteed.

3.3 Reliability

Disconnection between data plane and control plane due to fault tolerance may cause the network failure. So the reliability consideration is an important issue among multiple controllers in SDN. Hu discussed reliability in different topology and simulated the annealing metric. The reliability first go up and then go down as controllers numbers increases [26]. To manage the fault tolerant issues among controllers, a min-cut controller metric and heuristic metric compute the controller reliability in different topologies [25]. But still a good reliability metric required due to high cost increased by controllers.

3.4 Load Balancing

In distributed controller environment every switch is monitored by a controller in local domain. The communication traffic managed by controllers and as the network traffic increases the mapping of flow rules between controller and switch become overloaded and also few controllers become under loaded. Such imbalance degrades the performance of SDN network due to low throughput and high response time. Few metrics like, K-Center, Heuristic, cluster, greedy etc. are there to reduce the controllers load [25]. Due to overload, if controller become inactive or failed, the reassignment module triggered by above approaches to monitor and gather statics of controllers. There is a provisioning module which reassigns the controller and switch associations. In some cases the inactive controller may be deleted and a new controller may be added to manage the load and efficiency.

4 Analysis of Controller Placement Metrics on QoS Parameters

Hu [27] proposed a mechanism to take the decision for controllers placement which decides the required controllers in physical SDN networks. This mechanism maximized the resilience of a SDN network. Hock [28] suggested a POCO framework that reduced the latency and load balancing problems for controllers placements in SDN. Jimenez used the k critical algorithm to shade the loads of controller. This algorithm decides the numbers of controller required in network and also discover their locations to place them in SDN network [29]. Obadia [30] minimized the overhead of controllers and also optimized their locations for placements. In vehicular ad hoc network and in wireless sensor networks different independent nodes communicates in decentralized manner and having communication challenges on QoS parameters. [31–33]. Below the Fig. 3, depicted the feasibility of controller placement metrics on QoS parameters.

In the Table 3, we summarized and tabulated the different controller's performance based on certain QoS parameters e.g. reliability, scalability, consistency and load balancing.

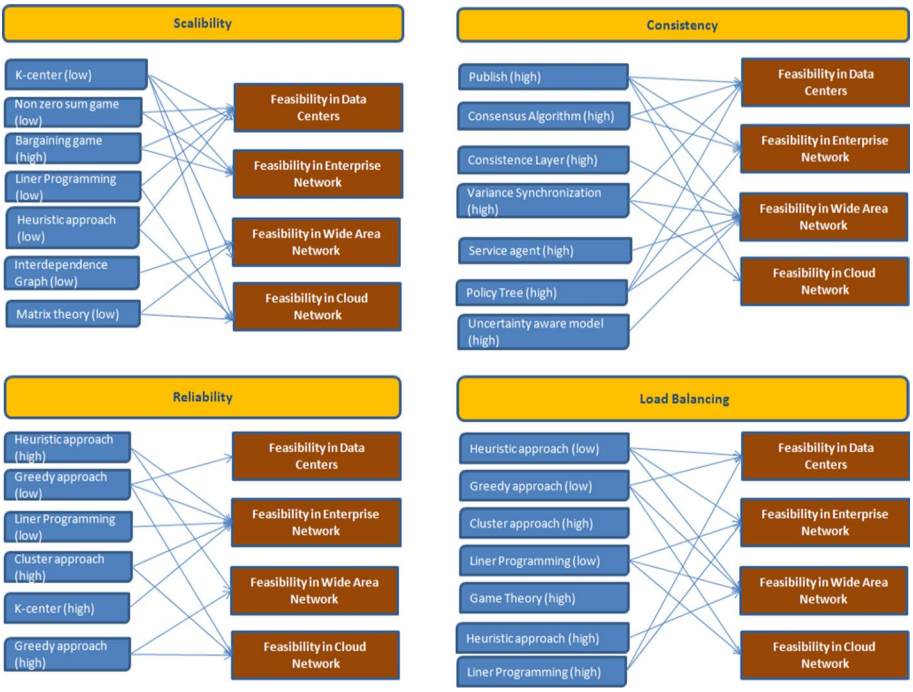

Fig. 3 Feasibility of Controller placement metrics on QoS parameters

4.1 Feasibility on Scalability

Heller tried to solves the controllers placements issues by scaling the SDN network. The authors give approach by using K-Center mechanism. By applying the K-Center method in simulation, Author found that as controller increases in network, the propagation latency is reduced [34, 35]. Rath suggested a distributed non-zerosum game mechanisms for optimal controllers placements in programmable network. By applying the Non zero sum game Author found that the simulation result reduces the packet delay and loss and also deployment cost while adding multi controllers [36]. Ksentini suggested a game model that takes multiple controllers into analysis and analyze the communication delays, processing loads and also optimize these problems. Author simulated the Bargaining game placement the controller method and summarized the result that the latency of controller communication with switch and other controller is minimized [37]. Sallahi and St-Hilaire suggested an approach to minimizes the costs of controllers placements which decides the numbers of controller required in network and also discover their locations to place them in SDN network. The simulation of Liner Programming method results that the designing of SDN network cost is reduced [38, 39]. Fu suggested a new plane i.e. Orion plane which reduced the complex computation for a wide SDN. To overcome the controller placement problem Heuristic and graph mechanism simulated. The Heuristic approach and graph theory approach results that the control plane complexity is reduced and the flow rate in this approach is better than Floodlight method [40, 41]. Wang used the NP Hard approach to reduce the number of controllers in large SDN networks. Author simulated the

Table 3 Comparative analysis of controller placement problem on QoS parameters

Sl. no.	Controller placement metrics	QoS parameters	Level of complexity	Feasibility in data centers	Feasibility in enterprise network	Feasibility in wide area network	Feasibility in cloud network	References
1	K-center	Scalability	Low	Y	Y	Y	Y	[34, 35]
2	Non zero sum game	Scalability	Low	Y	Y	N	N	[36]
3	Bargaining game	Scalability	High	Y	Y	N	N	[37]
4	Liner programming	Scalability	Low	Y	N	N	T	[38, 39]
5	Heuristic approach and graph theory	Scalability	Low	Y	N	N	Y	[40, 41]
6	Interdependence graph	Scalability	Low	N	N	Y	N	[42]
7	Matrix theory	Scalability	Low	N	N	Y	Y	[43]
8	Heuristic approach	Scalability	High	N	Y	N	N	[40, 41]
9	Publish	Consistency	High	Y	Y	Y	Y	[44]
10	Consensus algorithm	Consistency	High	Y	Y	N	N	[45]
11	Consistence layer	Consistency	High	N	N	Y	N	[46]
12	Variance synchronization	Consistency	High	Y	N	Y	Y	[47]
13	Service agent	Consistency	High	N	N	Y	N	[48]
14	Policy tree	Consistency	High	Y	Y	Y	N	[49]
15	Uncertainty aware model	Consistency	High	N	N	Y	N	[50]
16	Heuristic approach	Reliability	High	N	Y	Y	N	[26, 41]
17	Greedy approach	Reliability	Low	Y	Y	N	Y	[51, 52]
18	Liner programming	Reliability	Low	N	N	Y	N	[38, 39]
19	Cluster approach	Reliability	High	N	Y	N	Y	[53, 54]
20	K-center	Reliability	High	N	N	Y	N	[34, 35]
21	Greedy approach	Reliability	High	N	N	Y	Y	[51, 52]
22	Liner programming	Reliability	Low	N	N	Y	N	[39, 55]
23	Heuristic approach	Load balancing	Low	Y	Y	Y	Y	[26, 56]
24	Greedy approach	Load balancing	Low	Y	N	Y	Y	[51, 57]
25	Cluster approach	Load balancing	High	Y	N	Y	Y	[54]
26	Liner programming	Load balancing	Low	Y	N	N	Y	[39, 58]

Table 3 (continued)

Sl. no.	Controller placement metrics	QoS parameters	Level of complexity	Feasibility in data centers	Feasibility in enterprise network	Feasibility in wide area network	Feasibility in cloud network	References
27	Game theory	Load balancing	High	N	N	Y	Y	[59]
28	Heuristic approach	Load balancing	High	N	Y	N	N	[56, 60]
29	Liner programming	Load balancing	Low	Y	Y	N	N	[55, 61, 62]

Interdependence Graph method by using domain partitioning approach and gave the result that reduced the number of controllers in SDN [42]. Peng simulated the Matrix Theory in large SDN WAN where the large SDN network is partitioned and found that the throughput and latency performance improved. To validate the result, author suggested the Beacon frameworks for such implementation [43]. Caria suggested a hybrid SDN infrastructure so to partition the large-size networks among controllers. To partition the network distributed based routings mechanism used which sizes can be dynamically varied. By implementing the heuristic approach author concluded that the numbers of controllers are minimized and network capacity is improved [40, 41].

4.2 Feasibility on Consistency

Dontan implemented the Publish approach, and the result shown that while the network status changed then the controller's functionality remains same. This publish approach is based on WheelIFs distributed file systems which are three types. (a) Controller advertised itself in network i.e. control channels types. (b) data channels types where an event is publish and (c) individual controllers channels where any controller communicate individually to any other controller through command [44]. Ho suggested a Fast-Paxos-based-Consensus mechanisms which are based on consistencies approaches. The Consensus Algorithm is simulated and found that the controller's consistency is improved. In this mechanism the controllers are assigned a priority to handles the various switch requests [45]. Zhou considered the consistencies of controllers in SD-WAN and proposed a Consistence Layer. The placement metric Consistence Layer is simulated and result shows that the controllers cross domain complexity are reduced [46]. Guo proposed a Load-Varianc-based-Synchronization controller's state which improves the load-balance in SD-WAN. The placement metric Variance Synchronization is simulated and found that the synchronization overhead is reduced. In this mechanism first every controller should define their threshold and when limits exceeds the mechanism is applicable [47]. Phemius proposed the Distributed-Control-Planes for the heterogeneous topology networks which is used by various service agents. These are share the communications among controllers. This Service agent mechanism is simulated and found the result of proposed method enhance the inter domain consistency and reduced the delay [48]. Xiong proposed a framework which supports the customize consistent policy while networks are updating. Here is a hierarchical mechanism follows like a tree structure which use their individual packet forwarding principles. While SDN network is updating, the Policy Tree method provides the consistency if any link is failed [49]. This simulated in Mininet. Zhou suggested Uncertainty aware model which reduced the update time and gives more consistent network. Here the bandwidths and paths are considered to less overhead of memory [50].

4.3 Feasibility on Reliability

Hu suggested that the controller's reliability problems are NP-hard-problems. According to author by placing more than one controller when using Heuristic approach the packet delay time is reduced and enhances the network reliability [26, 41]. Beheshit proposed a connection metric for controller and switch in respect of distance factor to achieve high resilience. Authors enhanced the network reliability by using the greedy method for the proposed metric [51, 52]. Miiler used the Survivor mechanism which is a placement

method of controllers, to reduced the connection losses and enhance the survivability SDN networks. The probability of connection failure is found sixty percent after simulation Liner Programming methods [38, 39]. Song used the cluster approach method which improves the reliability quality by reducing the path distance in network. The reliability problems are discussed by the Control Path Managements frameworks [53, 54]. Jimenez developed a K-Critical algorithm to place the controller. This method identifies the numbers of required controllers and their placements locations in the networks. To maintain the reliability, K-center method locates the controller's positions and manages the network topology to reduce the latency [34, 35]. Sahoo proposed a method that takes the decision to place the controllers on a location so that the latency can be reduced and reliability can be improved. The Greedy method is simulated to achieve greater reliabilities in a control layer. In such type of implementation the transmission path is minimizes among controllers and switches. Hence the reliability is improved and the less number of controllers are able to manage large number of switches [51, 52]. Killi proposed a mathematical model to minimize the delays between controllers and switches. The limited numbers of controllers are placed in network according this model. Here the linear programming is simulated and as the number of traffic increased by multiple controllers the network congestion can be reduced by proposed method [39, 55].

4.4 Feasibility on Load Balancing

Hu used a clustering mechanism BalanceFlow and to reduce the loads of traffics in SDNs networks, the networks are partitioned and the heuristic method is simulated to reduce the recovery time [26, 56]. Selvi suggested a Cooperative Load Balancing mechanism to locate and place the controller. To balance the load of controllers the greedy method is used with suggested mechanism, To reduce the loads of traffic in SDN networks a hierarchical approach is followed and the greedy method is simulated to enhance the network throughput [51, 57]. Sufiev used the cluster approach to balance the load of controller in same cluster. To reduce loads of traffic in SDN networks a hierarchical approach is followed and cluster approach is simulated to balance the load dynamically [54]. Dixit suggested a switch-migration mechanism to monitor and balance the load of controller by switch-migration method. The switch is migrated from heavily loaded controller to light controller. To load balance the SDN network, the proposed method is simulated with liner programming mechanism and found that the response time of controller is minimized [39, 58]. Chen used the switch-migrations mechanism with game theory implementation. The switches are migrated from heavily loaded controllers to light controllers. The game theory is fast but not suitable for large SDN due to it's complexity. The proposed method is simulated to reduced the migration of switches and balance the network load [59]. Cheng suggested the switch-migrations mechanism based on available requests needs to serves by switches. To solve the switch-migrations to balance the loads a Heuristic approach and distributed hoping algorithm is simulated and reduced the flow-setup time to balance the network load [56, 60]. Yu proposed a load-informing approach in a distributed environment. This approach first measures the loads of switches, then inform the load, then takes the decision to balance and finally switch is migrated. The load information is shared periodically among controllers. The proposed method provides the better throughput and also balances the load with the implementation through liner programming approach [55, 61, 63].

Furthermore, the control layer is affected and overloaded by the placements of different controller because of increasing the network traffics among switches and controllers.

With the help of programmable network the efficiency and throughput of network can be improved.

5 Research Challenges Based on QoS in SDN

While research in SDN is growing the QoS is improving but still SDN deserves more research. Efficient controller placement and reliability of networks are biggest challenge which seeking more research effort. As per survey the research challenges to improve QoS in multiple controllers SDN networks are following:

5.1 Interaction Among Different Controllers and Switches

When multiple switches and controllers are placed in programmable networks domain, there may be different communication configuration implemented on devices. For the large scaled SDN, the multiple controller architecture is implemented. Different vendors may implements different communication policy on devices. Such policies generate the communication conflicts in the network.

5.2 Standard Protocol for Controller to Controller Communication

For the large network when multiple controllers are placed, a latency problem arises. There is no standard protocol for controller to controller communication on east–west control plane. To increase the throughput and reduce the latency such standard protocol is required. In a distributed environment the numbers of controllers are required and they directly affect the QoS of SDN network. To overcome this problem the different controller communication required in effective manner. To communicate controllers there are east–west interface is required. But here, is still needs to devise a protocol to standardize the east–west communication. Currently the global network supports the Border Gateway Protocol (BGP) for east–west interfaces.

5.3 Standard Protocol for Control Plane to Management Plane

For the northbound interface to interact control plane and management application plane, still needs to devise a protocol to standardize such communication. Due to such reason the overhead of communications increases due to different functionalities of different controllers. Currently SDN uses open source protocol to communicate control plane to management plane. For the large scaled multiple controllers, there are very difficult to scale the management interfaces. So a standard protocol in such communication may reduce the communication latency and may be help full in load balancing.

5.4 Dynamically Managing the Load Among Different Controllers

The communication traffic managed by controllers and as the network traffic increases the mapping of flow rules between controller and switch become overloaded and also few controllers become under loaded. Such imbalance degrades the performance of SDN network due to low throughput and high response time. Dynamically managing the load among different controllers is a big challenge. There is no standard mechanism for traffic engineering and load balancing. Different controllers may generate different traffic in the networks. While balancing the load controller's policies need to be change dynamically.

5.5 Controllers Security

As we know controller is a centrally monitoring and managing device therefore the security and protection of controller is required from any malicious attack. The malicious attacks may disrupt the control plane and data plane communication and also may be the reason of fault tolerance. Disconnection between data plane and control plane due to fault tolerance may cause the network failure. So the reliability consideration is an important issue among multiple controllers in SDN. So, this is also a research area to develop anti malicious attack mechanism.

5.6 Managing Heterogeneous Controllers

In distributed controllers due to architectural issues controller may impose inconsistence flow rules in switches and may result instability in programmable networks. The instability in network may cause service interruption, packet loss and asynchronous message communications among controllers. To maintain the uniformity of policies among various heterogeneous controllers in SDN is also very challenging. To avoid the performance interruption in this area, more research is required.

6 Conclusion

SDN is an attractive research area in today's computer networks communication area. But the challenge in this area is QoS in respect of controller management. For the large size network controller deployment is difficult to manage. The reliability and scalability are key challenges for QoS in programmable network. This paper investigates the programmable networks (SDN) overviews and briefed the QoS of SDN in respect of controller's challenges. SDN gives us the opportunity to solve the research towards Quality of Services. This paper also review and tabulated the available controller's summary. With the help of programmable network the efficiency and throughput of network can be improved. In this survey paper we analyzed the current research and summarized the findings of the different controller's performance based on certain QoS parameters e.g. reliability, scalability, consistency and load balancing. Finally, this paper concludes and highlighted some research challenges that may improve the QoS of SDN.

References

1. Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials, 16*(3), 1617–1634.
2. Singh, A. K., & Srivastava, S. (2018). A survey and classification of controller placement problem in SDN. *International Journal of Network Management, 28,* 1–25.
3. Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and Computer Applications, 67,* 1–25.
4. Alshnta, A. M., Abdollah, M. F., & Al-Haiqi, A. (2018). SDN in the home: A survey of home network solutions using Software Defined Networking. *Cogent Engineering, 5,* 1–40.
5. 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.
6. Wang, W., Tian, Y., Gong, X., Qi, Q., & Hu, Y. (2015). Software defined autonomic QoS model for future Internet. *The Journal of Systems and Software, 110,* 122–135.
7. Diego, F. M. V., Rothenberg, C., & Azodolmolky, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE, 103*(1), 14–76.
8. Wibowo, F. X. A., Gregory, M. A., Ahmed, K., & Gomez, K. M. (2017). Multi-domain software defined networking: Research status and challenges. *Journal of Network and Computer Applications, 87,* 32–45.
9. Erickson, D. (2013). The Beacon OpenFlow controller. In *Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, ser. Hot SDN'13*. New York, NY, USA: ACM (pp. 13–18).
10. Tootoonchian, A., & Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. In *Proceedings of 2010 internet network management conference on Research on enterprise networking* (p. 3–3). USENIX Association.
11. Cai, Z., Cox, A. L., & Ng, T. S. E. (2011). *Maestro: A system for scalable OpenFlow control*. Rice University, Tech. Rep.
12. Krishnaswamy, U., Berde, P., Hart, J., Kobayashi, M., Radoslavov, P., Lindberg, T., Sverdlov, R., Zhang, S., Snow, W., & Parulkar, G. (2013). *ONOS: An open source distributed SDN OS*. http://www.slideshare.net/umeshkrishnaswamy/open-network-operating-system.
13. Saikia, D. (2013). *MuL OpenFlow controller*. http://sourceforge.net/projects/mul/.
14. Nippon Telegraph and Telephone Corporation. (2012). Ryu Network Operating System. http://osrg.github.com/ryu/.
15. McCauley, M. (2012). POX. http://www.noxrepo.org/.
16. Juniper Networks. (2013). Opencontrail. http://opencontrail.org/.
17. OpenDaylight. (2013). OpenDaylight: A Linux Foundation Collaborative Project. http://www.opendaylight.org.
18. The Real-Time Cloud. (2014). http://www.ericsson.com/res/docs/whitepapers/wp-sdn-and-cloud.pdf.
19. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). *NOX: Towards an operating system for networks*. Comp. Comm. Rev.
20. HP VAN SDN Controller Software. http://h20195.www2.hp.com/v2/getpdf.aspx/4AA4-9827ENW.pdf.
21. Floodlight is a Java-based OpenFlow controller. (2012). http://floodlight.openflowhub.org/.
22. Kopsonen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., et al. Onix: A distributed.
23. Zhang, Y., Cui, L., Wang, W., & Zhang, Y. (2018). A survey on software defined networking with multiple controllers. *Journal of Network and Computer Applications, 103,* 101–118.
24. Karakus, M., & Durresi, A. (2017). A survey: Control plane scalability issues and approaches in Software-Defined Networking. *Computer Networks, 112,* 279–293.
25. Changhe, Yu., Lan, J., Xie, J., & Yuxiang, H. (2018). QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs. *Procedia Computer Science, 131,* 1209–1216.
26. Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2014). On reliability-optimized controller placement for software-defined networks. *China Communications, 11*(2), 38–54.
27. Hu, Y. N., Wang, W. D., Gong, X. Y., Que, X. R., & Cheng, S. D. (2012). On the placement of controllers in software-defined networks. *Journal of China Universities of Posts and Telecommunications, 19*(Supplement 2), 92–171. https://doi.org/10.1016/S1005-8885(11)60438-X.

28. Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., & Tran-Gia, P. (2013). Pareto- optimal resilient controller placement in SDN-based core networks. In *Teletraffic congress (ITC), 25th international* (pp. 1–9) https://doi.org/10.1109/itc.2013.6662939.
29. Jimenez, Y., Cervello-Pastor, C., Garcia, A. (2014). On the controller placement for de-signing a distributed SDN control layer. In *Networking conference, 2014 IFIP* (pp. 1–9). https://doi.org/10.1109/ifipnetworking.2014.6857117.
30. Obadia, M., Bouet, M., Rougier, J.L., Iannone, L. (201). A greedy approach for minimizing SDN control overhead. In *2015 1st IEEE conference on network Softwarization (NetSoft)* (pp. 1–5). https://doi.org/10.1109/netsoft.2015.7116135.
31. Pandey, P., Kansal, V., & Swaroop, A. (2020). Vehicular ad hoc networks (VANETs): Architecture, challenges and applications. In *Handling priority inversion in time-constrained distributed databases* (pp 224–239). IGI Global. https://doi.org/10.4018/978-1-7998-2491-6.ch013.
32. Gupta, S., Rana, A., & Kansal, V. (2020). Optimization in wireless sensor network using soft computing. In *Proceedings of the third international conference on computational intelligence and informatics. Advances in intelligent systems and computing*, *1090*, 801–810. https://doi.org/10.1007/978-981-15-1480-7_74.
33. Pandey, P., Kansal, V., & Swaroop, A. (2019). A concise survey on recent routing protocols for vehicular adhoc networks (VANETs). In *Proceedings IEEE 2019 international conference on computing, communications, and intelligent systems (ICCCIS)* (pp. 188–193), IEEE Explore. https://doi.org/10.1109/icccis48478.2019.8974464.
34. Heller, B., Sherwood, R., & McKeown, N. (2012). The controller placement problem. In *Proceedings of 1st workshop HotSDN* (pp. 7–12).
35. Jimenez, Y., Cervello-Pastor, C., & García, A. J. (2014). On the controller placement for designing a distributed SDN control layer. In *Proceedings of IFIP networking conference*, Trondheim, Norway (pp. 1–9).
36. Rath, H. K., Revoori, V., Nadaf, S. M., & Simha, A. (2014). Optimal controller placement in Software De ned Networks (SDN) using a non-zero-sum game. In *Proceedings of IEEE international symposium on world wireless, mobile multimedia networking*, Sydney, NSW, Australia (pp. 1–6).
37. Ksentini, A., Bagaa, M., Taleb, T., & Balasingham, I. (2016). On using bargaining game for Optimal Placement of SDN controllers. In *Proceedings of IEEE international conference communication (ICC)* (pp. 1–6), Kuala Lumpur, Malaysia, May 2016.
38. Sallahi, A., & St-Hilaire, M. (2015). Optimal model for the controller placement problem in software de ned networks. *IEEE Communications on Letters, 19*(1), 30–33.
39. Müller, L. F., Oliveira, R. R., Luizelli, M. C., Barcellos, M. P., & Gaspary, L. P. (2014). Survivor: An enhanced controller placement strategy for improving SDN survivability. In *Proceedings of IEEE global communication conference*, Austin, TX, USA (pp. 1909–1915).
40. Fu, Y., et al. (2015). A hybrid hierarchical control plane for ow-based large-scale software-defined networks. *IEEE Transactions on Network and Service Management, 12*(2), 117–131.
41. Caria, M., Jukan, A., & Hoffmann, M. (2016). SDN partitioning: A centralized control plane for distributed routing protocols. *IEEE Transactions on Network and Service Management, 13*(3), 381–393.
42. Wang, G., Zhao, Z., Peng, J., & Li, R. (2014). An approximate algorithm of controller configuration in multi-domain SDN architecture. In *Proceedings of 9th international conference on communication networks*, China, Maoming (pp. 601–605).
43. Xiao, P., Li, Z.-Y., Guo, S., Qi, H., Qu, W.-Y., & Yu, H.-S. (2016). AK self-adaptive SDN controller placement for wide area networks. *Frontiers of Information Technology & Electronic Engineering, 17*(7), 620–633.
44. Dotan, D., & Pinter, R. Y. (2005). `HyperFlow: An integrated visual query and dataflow language for end-user information analysis. In *Proceedings of IEEE Symposium on visual languages and* human-centric computing *(VL/HCC)* (pp. 27–34).
45. Ho, C. C., Wang, K., & Hsu, Y. H. (2016). A fast consensus algorithm for multiple controllers in software-defined networks. In *Proceedings of international conference on advanced communication technology* (p. 1).
46. Boyang, Z., Chunming, W., Wen, G., Hong, X., Jiang, M., & Shuangxi, C. (2015). Achieving consistence for cross-domain WAN control in software defined networks. *China Communications, 12*(10), 136–146.
47. Guo, Z., et al. (2014). Improving the performance of load balancing in software defined networks through load variance-based synchronization. *Computer Networks, 68,* 95–109.

48. Phemius, K., Bouet, M., & Leguay, J. (2014). DISCO: Distributed multi-domain SDN controllers. In *Proceedings of IEEE network operations and management symposium (NOMS)*, Krakow, Poland, May 2014 (pp. 1–4).
49. Xiong, X., & Fu, J. (2011). Active status certificate publish and subscribe based on AMQP. In *Proceedings of international conference* on *computer science*, Chengdu, China (pp. 725–728).
50. Zhou, W., & et al. (2015). Enforcing customizable consistency properties in software-defined networks. In *Proceedings of Usenix conference network systems design implement*. USENIX Association (pp. 73–85).
51. Beheshti, N., & Zhang, Y. (2012). Fast failover for control traffic in software defined networks. In *Proceedings of IEEE global communication conference (GLOBECOM)*, Anaheim, CA, USA (pp. 2665–2670).
52. Sahoo, K. S., Sahoo, B., Dash, R., & Jena, N. (2016). Optimal controller selection in software de ned network using a greedy-SA algorithm. In *Proceedings of IEEE conference Indiacom* (pp. 2342–2346).
53. Song, S., Park, H., Choi, B.-Y., Choi, T., & Zhu, H. (2017). Control path management framework for enhancing software-defined network (SDN) reliability. *IEEE Transactions on Network and Service Management, 14*(2), 302–316.
54. Sufiev, H., & Haddad, Y. (2016). A dynamic load balancing architecture for SDN. In *Proceedings of IEEE international conference on science electrical engineering (ICSEE)*, Eilat, Israel (pp. 1–3).
55. Killi, B. P. R., & Rao, S. V. (2016). Optimal model for failure foresight capacitated controller placement in software-defined networks. *IEEE Communications Letters, 20*(6), 1108–1111.
56. Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2013). BalanceFlow: Controller load balancing for OpenFlow networks. In *Proceedings of IEEE international conference on cloud computing intelligent systems* (pp. 780–785).
57. Selvi, H., Gür, G., & Alagöz, F. (2016). Cooperative load balancing for hierarchical SDN controllers. In *Proceedings of IEEE 17th international conference on high performing switching routing (HPSR)*, Yokohama, Japan (pp. 100–105).
58. Dixit, A., Hao, F., Mukherjee, S., Lakshman, T. V., & Kompella, R. R. (2014). ElastiCon; an elastic distributed SDN controller. In *Proceedings of ACM/IEEE symposium architecture networking communication systems (ANCS)*, Marina del Rey, CA, USA (pp. 17–27).
59. Chen, H., Cheng, G., & Wang, Z. (2016). A game-theoretic approach to elastic control in software-defined networking. *China Communications, 13*(5), 103–109.
60. Cheng, G., Chen, H., Wang, Z., & Chen, S. (2015). DHA: Distributed decisions on the switch migration toward a scalable SDN control plane. In *Proceedings of IFIP networking conference (IFIP)* (pp. 473–477).
61. Yu, J., Wang, Y., Pei, K., Zhang, S., & Li, J. (2016). A load balancing mechanism for multiple SDN controllers based on load informing strategy. In *Proceedings of Networking Operation Management Symposium* (pp. 1–6).
62. Song, P., Liu, Y., Liu, T., & Qian, D. (2017). "Controller-proxy: Scaling network management for large-scale SDN networks. *Computer Communications, 108,* 52–63.
63. Shih-Chun Lin, P., & Wang, M. L. (2016). Control traffic balancing in software defined networks. *Computer Networks, 106*, 260–271.

Surendra Kumar Keshari obtained his Master in Computer Application degree from Uttar Pradesh Technical University Lucknow, India, in 2007 and M.S. degree in Cyber Law and Information Security from Indian Institute of Information Technology, Allahabad, India in 2009 and currently a Ph.D. student in, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India. He is working as assistant professor in KIET Group of Institutions Ghaziabad, India. His research interest is Computer Network, Software Defined Network and Information Security.

Vineet Kansal studied at Indian Institute of Technology, Delhi and is currently working as Professor with Institute of Engineering and Technology, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India. He was awarded appreciation by NPTEL, IIT Kanpur and Centre of Continuing education, IIT Kanpur for inspiring the faculty members and students of higher technical education to adopt NPTEL Online certification courses, for evangelizing it's modus operandi and for conceptualizing online and offline blended Faculty training programs addressing pedagogical issues in engineering education in the state of Uttar Pradesh, India.

Sumit Kumar is currently working as associate Professor in Department of Computer science and engineering, Amity school or engineering and technology, Amity University Noida Utter Pradesh India. He did his Ph.D. degree from NIT Jamshedpur Jharkhand India. He has more than 20 years of teaching and research experience at repute college and university of India. His area of research is software testing, meta-heuristic algorithms image processing. He has published more than 25 international journals and conferences of repute. He is the organizing chair of 11th international conference confluence 2021. He is also the review of many international journals and conferences.