

A SYSTEMATIC LITERATURE REVIEW ON QUALITY OF SERVICE (QOS)
IN MULTI CONTROLLER'S LARGE SIZE SOFTWARE DEFINED
NETWORKS (SDN)

Mahar Muhammad Aown

DOI: <https://doi.org/10.5281/zenodo.23364316>

Keywords

SLR, Multi-Controller, SDN, QoS, Networks, Networking, QoS in Multi-Controller

Article History

Received: 25 June 2026

Accepted: 16 September 2026

Published: 14 October 2026

Copyright @Author

Corresponding Author: *

Mahar M Aown

Abstract

Network infrastructure paradigm, Software-Defined Networking (SDN), allows networks to be more easily managed. The single centralized controller is unable to handle the increased flow processing requirements as the network grows in size. As a result, the multi-controller appears to be a potential option for SDN in big networks. An SDN compressive survey for multiple controllers is presented in this study. In the beginning, we give a general review of multi-controller, including its origins and problems. In addition, we have discussed different perspectives such as the architecture of SDN, SDN controllers and evolution, Relationship between SDN and QoS. In the last, different opportunities and challenges have been deliberated.

INTRODUCTION

An interconnected network of computers is called a computer network. There are numerous intermediary nodes in the network, such as repeaters and switches. Computer networks of the future are distributed, packet-switched networks that may be accessed from nearly any digital device on the planet. It's tough to maintain a vast network, because regulations for forwarding traffic are constantly changing on a typical network. The idea of a programmable network was born out of a desire to improve

upon the limitations of the traditional network. Software-defined networks (SDNs) manage the behavior of programmable networks (SDN). Unlike traditional networks, this dynamic network-infrastructure maintains the data plane and control plane independently. It is conceptually independent and centralized to manage the control plane, while the forwarding data plane follows the instructions of the control plane. Figure-1 depicts the SDN architecture [1],[2].

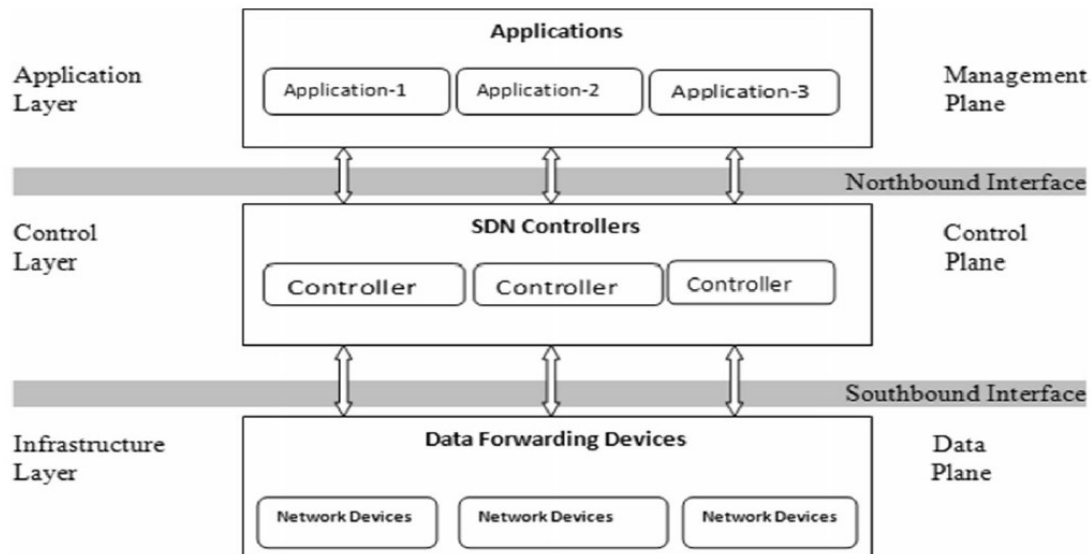


Figure 1 SDN Architecture

To establish computer networks, network devices such as routers, switches, and different middle box devices (i.e. devices that control traffic for reasons other than packet forwarding, such as a firewall) are commonly utilized. In order to deal with the large variety of network events and applications, a wide variety of rules are required. Translating these high-level rules into low-level configuration commands by hand is necessary for adapting to changing network conditions. It's not uncommon for them to have to complete these extremely difficult tasks with only a few rudimentary tools. To make matters even more difficult, network management and performance tuning is error-prone as a result. The difficulties faced by network administrators and operators are exacerbated by the prevalence of vertically integrated black boxes. "Internet ossification" has been labelled as another nearly insurmountable problem facing network professionals and researchers. The Internet's physical infrastructure, protocols, and performance have all become extremely difficult to improve due to the Internet's large installed base and status as a critical component of our society's infrastructure (along with transportation and power grids). If present and future internet applications and services grow increasingly complicated and

demanding, it is imperative that the internet change to meet these new difficulties.

When it comes to networking, Software-Defined Networking (SDN) is a new paradigm in which control decisions are decoupled from forwarding hardware. Simplifying network management and allowing for new ideas are two of the main benefits it promises. Network resources may now be relied upon by software developers in the same way that storage and compute resources can. Network equipment are reduced to basic packet forwarders (the data plane) that may be configured via an open interface in order to achieve software-defined networking. Software-defined networking (SDN) is a term for this concentration of network intelligence (e.g., ForCES, OpenFlow, etc) [3], [4].

SDN is now receiving a lot of interest from the academic community as well as the business community. Recently, an industrial-driven organization, the Open Network Foundation (ONF), has been formed to promote SDN and standardize the OpenFlow protocol [2]. A research Centre dedicated to SDN has been established at the OpenFlow Network Research Center [4]. The IETF, IRTF, and other standards bodies have also been involved in SDN standardization efforts [4].

Routers and switches, which use distributed control and transport network protocols, allow digital packets of information to go around the world. In spite of their extensive use, traditional IP networks are complex to manage. In order to describe desired network regulations, each network device must be configured using low-level and typically vendor-specific instructions. Besides the complexity of configuration, networks have to deal with the dynamic faults and changes in load. Current IP networks do not have any mechanisms for automatic reconfiguration or response. Because of this, enforcing policies in such a fluid environment is extremely difficult. Vertical integration further complicates things in the current network environment. Networking equipment combine the control and data planes, which reduces flexibility and inhibits network infrastructure innovation and growth. IPv4 to IPv6 migration began more than a decade ago and is still mainly unfinished, proving the complexity of the change. Since the existing IP networks are inert, it can take anywhere from five to 10 years to design, analyze, and implement a whole new routing system. A clean-slate strategy, like modifying the Internet's design (for example, by replacing IP), is viewed as unfeasible to put into practice. As a result, the costs of setting up and maintaining an IP network have gone through the roof [5], [6], [7].

The traditional network's weaknesses can be addressed by using Software-Defined Networking (SDN). SDN is a paradigm shift in network technology that undermines the fundamental assumption of conventional networks. An SDN consists of a data plane, a control plane, and an application plane. Sending data to the data plane is done via the control plane, which is comprised of network equipment (such as a router and switch). Data and application planes communicate via the control plane, which serves as an intermediary between the two. Application logic, such as detecting intrusions or conducting large-scale data analyses, is housed on top of the control plane [8], [9], [10].

In the control plane's preliminary architecture, each network has only one controller. Despite the benefits of SDN's centralized management, the ever-increasing network needs put pressure on SDN's nature (i.e. centralized control). Even though network operators try their best to increase network controller performance, it is still difficult to meet the high demands (e.g., flow requests made by switches and network statistics) due to the limited capacity of a single controller. It is limited to 6000 packets per second, with an average latency of less than a millisecond for Ryu (6ms). Networks of a large size are especially prone to this issue. It is also important to address the single point of failure in an SDN network with a single controller [11].

2. RESEARCH METHODOLOGY

SLR (systematic literature review) guidelines from software engineering have been used in our research project. As part of the SLR, existing studies are systematically reviewed in a repeatable and unbiased way, with several steps for identifying, analyzing, and interpreting all relevant current information. Figure-1 depicts our research's overall progress.

2.1 RESEARCH QUESTIONS

Before beginning the process of looking for and selecting relevant articles, we developed a series of research questions. The research questions serve as a foundation for further processes in the review process, such as data extraction, inclusion/exclusion criteria, and search string design. The following are the clearly stated research concerns: **Quality of Service (QoS) in Multi Controller's Large Size Software Defined Networks (SDN)**

RQ 1: What is the Multi-controller and its challenges?

RQ 2: What are the architectures of SDN controllers along with its evolution? RQ 3: What is the relationship between SDN and QoS?

RQ 4: What are the main challenges and opportunities in QoS in Multi-controller large size SDN?

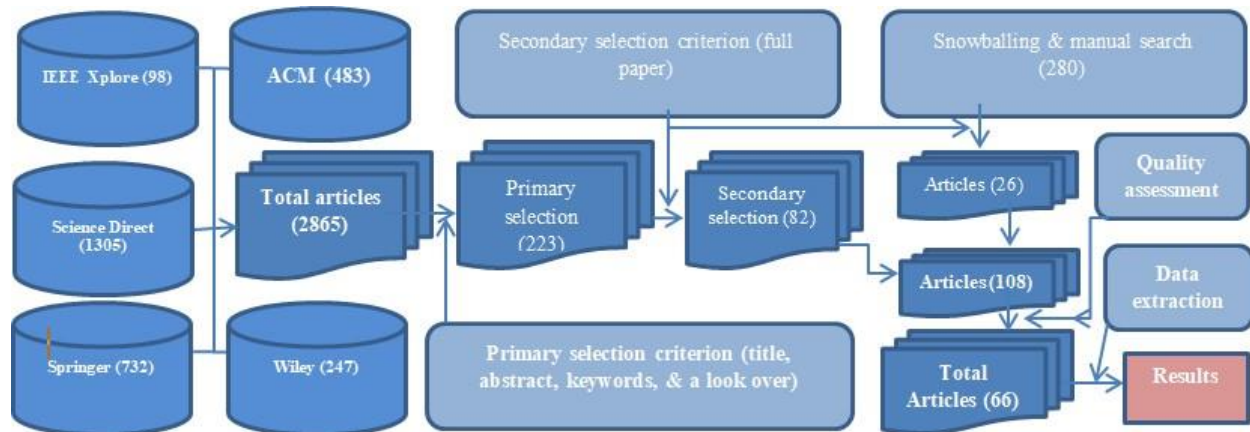


Figure-1 Research methodology, including search strategy, article selection criteria, quality assessment, and data extraction. Research articles were found in the following libraries.

- 1) IEEE Xplore <http://ieeexplore.ieee.org>,
- 2) ACM digital library <https://dl.acm.org>,
- 3) ScienceDirect <http://www.sciencedirect.com>,
- 4) SpringerLink <https://link.springer.com>, and
- 5) Wiley <https://onlinelibrary.wiley.com/>

Data extraction

Our research questions are based on the current state of the art Quality of Service (QoS) in Multi Controller's Large Size Software Defined Networks (SDN), as well as those of the selected articles' Multi Controller's Large Size SDN.

2.2 Inclusion and exclusion criteria

We began by looking for publications based on their abstracts, keywords, and titles. As part of the second phase, we'll add articles on Quality of Service (QoS) in Multi Controller's Large Size Software Defined Networks (SDN) such as current state-of-the-art, possibilities and problems, and methods and tools.

We also didn't include publications that weren't peer-reviewed, were written in a language other than English, were duplicates of other articles, or were otherwise of low quality.

3. LITERATURE REVIEW

3.1 SDN Controllers

At the upper layer of the programmable network, RYU is a controller for SDN that logically

manages APIs for various applications. RYU has its sights set on cloud and data Centre environments. Traffic management is supported in RYU ver1.0, RYU ver1.2, and RYU ver1.3. NTT cloud data centers created it in Python. Java-based floodlight controllers are Beacon-compliant and can control both real and phantom-powered switches. On a scale of 12 threads, these controllers can handle up to 1.39 million requests per second. For large data centers and high-performance machines, this controller is primarily designed. Their own northbound APIs were also defined. Floodlight, like several other controllers, undermines security by changing data, which compromises the accuracy of the data. A downside of POX is that it responds slowly. There are security concerns with this controller due to the fact that any data change compromises the data's reliability. This controller may be used with Python. POX systems make it simple to operate OpenFlow. OSGi Frameworks are supported for controllers in the autonomous system by OpenDaylight Java, which also supports REST in the North bound API. The CLI and GUI versions of these controllers are both readily available. RESTCONF and NETCONF are only two of the many models that make up the foundation of OpenDaylight. OpenDaylight provides a scalability and performance-enhancing configuration framework for SDN networks. REST and JAVA APIs are

among the northbound APIs supported. With a SAL, OpenDaylight divides its APIs into two groups: northbound and southbound (Service Adaption Layer). OpenFlow, OVSDDB, PCEP, BGP, SNMP, and other southbound APIs are supported. Implements and supports OpenDaylight with its programmable network controller. It was initially a single-threaded controller called NOX, but it has since been updated to allow a multithreaded environment known as NOX-MT. It is capable of efficiently handling a large number of requests with rapid input. In order to forward the packet, it uses a hash table to match up the MAC and port numbers on the switch. New MAC addresses are added to the hash table, which allows for read operations. Despite the fact that Hyperflow is a centralized controller, it is distributed. By enabling connectivity with other dispersed controllers, it lowers administrative costs. It was created using NOX design. Control of each switch is handled by a separate controller's software and controllers send a message to the network every few minutes indicating their presence. SDN controllers that provide network scalability in the WAN and are open source are called ONOS (Open Network Operating Systems). Using distributed controllers, the network's performance, scalability, and availability concerns may be managed. About one million

flows per second can be handled by ONOS. This is a centralized controller capable of worldwide network monitoring and performance enhancement. Using SDN technology, the OpenContrail controller mixes protocols like as BGP, SNMP, and others. Both the CLI and GUI versions of these controllers support the northbound API. Using OpenDaylight, Ericsson SDN controllers tie end-user service customization inside networks to Ericsson SDN controllers. As a result of HP VAN's use of OpenDaylight as its foundation, these SDN controllers offer OpenFlows protocols and cluster implementation for high-availability, multi-application environments. These protocols include BGP and SNMP, as well as SDN

technologies. This is an open source controller. OpenFlow, L3 Agent, and L2 Agent are just a few of the southbound APIs that these controllers offer. Onix is a decentralized yet conceptually centralized controller. By allowing contact with local controllers, it lowers the administrative burden. Control of traffic and routing as well as network scalability are all made easier with this device. System failures are handled by this multi-language controller, such as C++, JAVA and Python. Fault tolerance and data consistency concepts are supported by Onix controllers. OVSDDB and OpenFlow are two examples of southbound APIs that Onix supports. A general overview of SDN controllers may be shown in Table-1 [12-27].

Table-1(SDN Controllers)

Sl no.	Controller name	Organization	Program- ming language	Open source	Flows/ second	Modularit y	Productivit y	Consistenc y	Fault	Architecture
1	Beacon	Stanford Univer- sity	Java	Yes	12.8 M	Fair	Fair	No	No	Centralized Multi- threaded
2	Hyperflow	University of Toronto	++C		30 K	Fair	Fair	Weak	Yes	Distributed
3	Maestro	Rice Univ.	Java	Yes	4.8 M	Fair	Fair	No	No	Centralized Multi- threaded
4	ONoS	ON.Lab	Java	Yes	1 M	High	Fair	Weak, strong	Yes	Distributed
5	OpenMUL	KulCloud	C	Yes				No	No	Centralized Multi- threaded
6	RYU	NTT	Python	Yes		Fair	Fair	No	No	Centralized Multi- threaded
7	POX	Nicira	Python	Yes	1.8 M	Low	Fair	No	No	Centralized
8	OpenContrail	Juniper	Python	Yes		Fair	Fair	No	No	Distributed
9	OpenDaylight	Linux Foundation	Java	Yes	106 K	High	Fair	Weak	No	Centralized
10	Ericsson SDN Controller	Ericsson	Java	No						Distributed
11	NOX	Nicira	Python	Yes	1.8 M	Low	Fair	No	No	Centralized
12	HP VAN SDN Controller	HP	Java	No						Distributed
13	FloodLight	Big Switch Net- work	Java	Yes		Fair	Fair	No	No	Centralized multi- threaded
14	Onix	Nicira Networks	C, Python	Yes	2.2 M	Fair	Fair	strong	Yes	Distributed

MULTI-CONTROLLER OVERVIEW CONTROLLER EVOLUTION

It is in this section that we begin by introducing the origin and basic architecture of multi-controllers, using examples of flat and hierarchical designs.

1) FROM SINGLE CONTROLLER TO MULTICONTROLLER

It is at this stage that a single controller manages the network. It is shown in Figure-2 that a controller (c1) is responsible for four switches in the network. New packets cannot be forwarded by switch (s1) because of a lack of routing information in new packets from the source

host. Next, the switch (s1) communicates with the controller (c1) to obtain the routing information for a new packet. The switch forwards the packet to the next device after receiving a response message from the controller. Finally, the packet is successfully received by the intended host. The controller has a significant impact on the flow of traffic. Sadly, as network traffic grows, a single controller's limited capacity prevents it from handling the large number of flow requests sent by switches. Conversely, switches are unable to design packet routing when a single controller fails, which disrupts network communication and application functionality. Consequently, a new controller

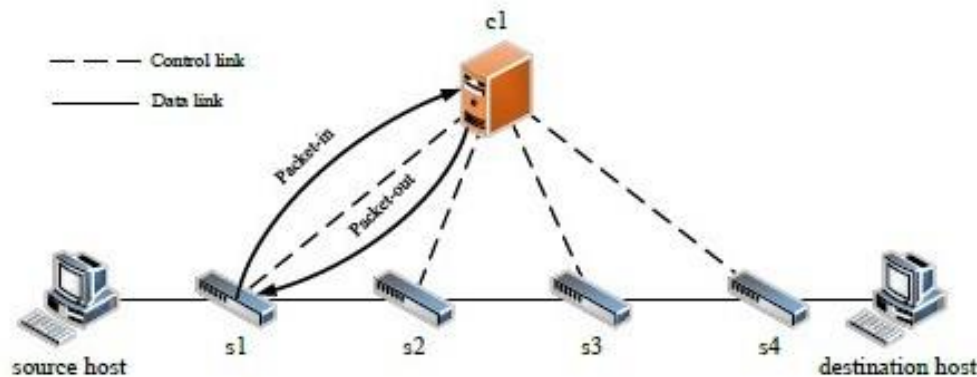


Figure 2 Single Controller Working

Using OpenFlow (e.g., the concept of a master controller linking to many slaves or equal controllers via a single switch), the multi-controller architecture provides a revolutionary SDN strategy that may overcome the single controller dilemma. The network is separated into two sections, each under the direction of a separate network controller. All related switches can immediately build forwarding pathways for packets that arrive at (s1)

using the same centralized logic as (c1) and (c2). For example, a single flow- processing controller may be successfully eased in this manner. There are two controllers that can help each other out in the meanwhile, which might address the controller's sole failure point. A multi-controller SDN network is more efficient than a single-controller SDN network, as seen above. Because of this, multi-controller systems have been increasingly common in the last several years [28].

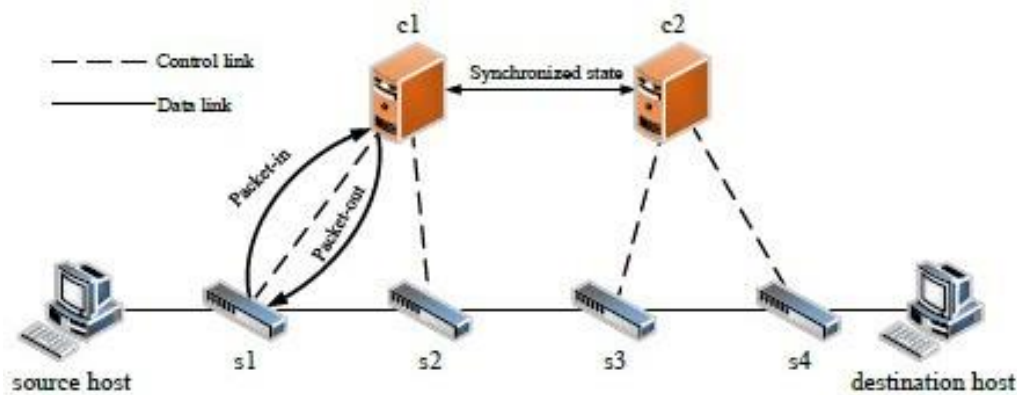


Figure 3 Multi-Controller Working

2) TWO BASIC MULTI-CONTROLLER ARCHITECTURES

In multi-controller systems, the most important consideration is how to design the architecture. Flat design and hierarchical design are two of the most common types of multi-controller architecture. Networks in flat design are divided into smaller, self-contained sections, each of

which is managed by a central controller located in a separate local network view. Using their east-west interfaces, controllers are able to see the whole network. Figure 4 depicts the multi-controller flat design. HyperFlow and Onix are two common examples. HyperFlow is built for WheelFs, a distributed file system that runs on NOX. Only flow requests from switches in the

same domain are processed by each controller in HyperFlow. Information on network traffic and routing is exchanged in a "publish/subscribe" style between controllers. As part of Onix's distributed design, the top control logic may be accessed via a programmatic interface and the global network state is maintained via the Network Information Base. Onix receives network status information from the physical infrastructure and uses the connectivity

infrastructure to carry out its operations. Having a flat architecture increases the control plane's potential but comes at the cost of more complex controller management and more control overheads. To ensure a consistent network view, the controllers, for example, must often communicate with one another. Hierarchical organization is suggested as a means of resolving the issues [28], [29], [30], [31].

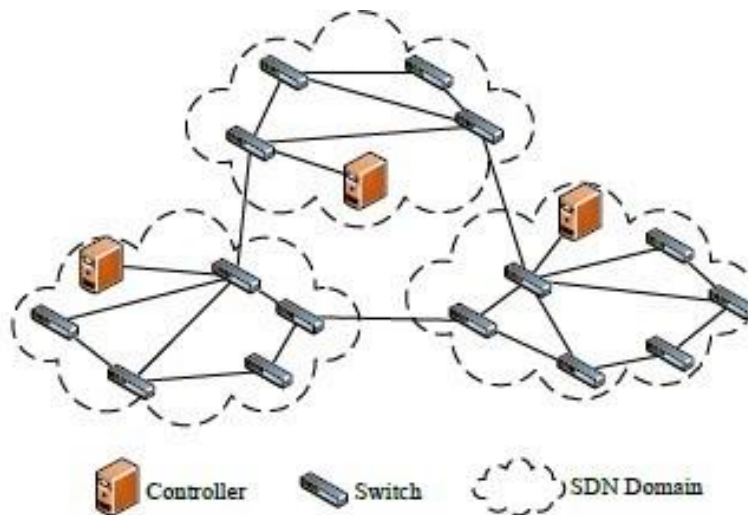


Figure 4 Multi-controller Design

Root and domain controllers are typically employed in a hierarchical network design. The root controller is responsible for overseeing all of the other controllers within a given domain as well as running local control programs. The

Kandoo controller hierarchy is a standard practice. Rather of relying on the communication between domain controllers, the Kandoo's root controller does this task. Figure 5 shows the basic framework of a hierarchical design [32].

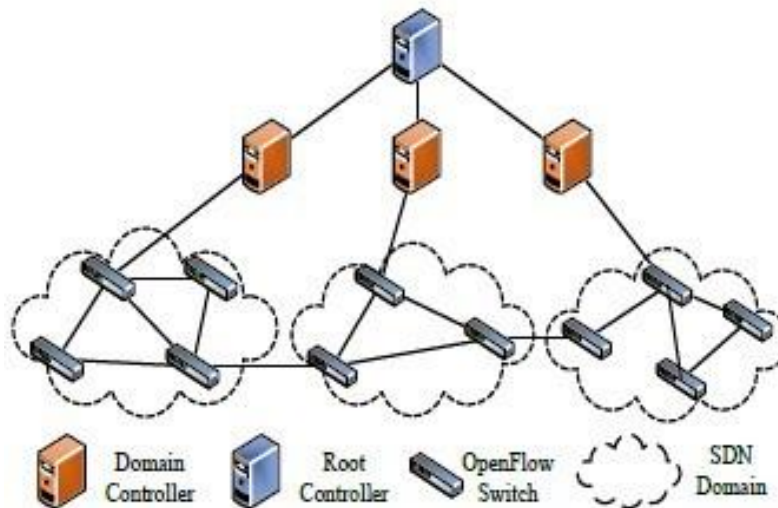


Figure 5 Architecture of Hierarchical Design

Relationship between SDN and QoS

It's common to think of QoS as the capacity of a network to deliver essential services to traffic. Priority is given to QoS parameters, such as, but not limited to, the following:

- ✓ bandwidth
- ✓ delay
- ✓ jitter
- ✓ loss

To provide quality of service (QoS), application flows must be differentiated since they compete for network resources. As a result, the proper allocation of network resources must give precedence to the higher-priority traffic. The proper packet forwarding decisions typically need knowledge of the existing network conditions during this procedure. End users and service providers are relying on Service Level Agreements (SLAs) to offer Quality of Service (QoS). For best effort service, this method works well. However, it does not provide pergranulartrac control. But there are other kinds of applications that are sensitive to the latency, the jitter, and the bandwidth, such as VoIP, online-gaming and

video conferencing. It can also be difficult to monitor the Internet's hop-by-hop" decision architecture, particularly due of the numerous different vendor-specific software sat use. High-level trac control policies and constraints on the depth of trac differentiation cannot be specified in a uniform manner. Hard QoS and soft QoS are the most common methods for implementing QoS. The hard QoS approach ensures the connection's QoS criteria, but its user is limited in resources. QoS-guaranteeing methods like IntServ are examples of this approach. In contrast to hard QoS approaches, the soft QoS method does not have the same severe standards for QoS compliance. Soft QoS methods, such as Di Serv, are an example. For the purposes of this study, we used two types of Quality of Service (QoS) models: hard (hard QoS) and soft (soft QoS). In the research, several measures are designated as target QoS metrics. As a result, the QoS measures in Table 2 show a wide range of issues that have been addressed in the research [33-55].

Table-2

Techniques	QoS Models							
	Bandwidth	Hard QoS Delay	Jitter	Loss	Bandwidth	Soft QoS Delay	Jitter	Loss
Wallner et al.					X	X		
Civanlar et al.					X	X		X
HiQoS					X	X		
OpenQoS					X	X		X
VSDN	X	X	X					
RVSDN	X	X	X					
Tomovic et al.	X							
Egilmez et al.					X	X		X
Egilmez et al.					X	X	X	X
ARVS							X	X
Yilmaz et al.							X	X
Egilmez et al.							X	X
Egilmez et al.					X	X	X	X
Karakus et al.	X	X						
FlowBroker	X	X		X	X	X		X
Wang et al.	X	X						
Miao et al.		X		X		X		X
CXP	X	X			X	X		
Kim et al.	X	X		X				
NCL	X	X	X					
Duan et al.	X	X			X	X		
FlowQoS	X	X			X	X		
Afaq et al.					X			
QoSFlow					X			
OpenQFlow	X				X			
Xu et al.					X	X	X	
Wang et al.	X	X			X	X		
Caba et al.	X	X			X	X		
Truong et al.					X	X	X	X
Kumar et al.	X				X			
Yiakoumis et al.	X	X	X		X	X	X	
Kassler et al.					X	X		
Q-POINT	X	X		X	X	X		X
QFF					X			
Gorlatch et al.					X	X		
Jarschel et al.	X				X			
Ayadi et al.					X	X		
Q-Ctrl	X				X			
PolicyCop					X	X		
OpenCache					X	X		
Sonkoly et al.	X	X	X		X	X	X	
SoIP					X			
ACDPA					X	X		

CHALLENGES AND OPPORTUNITIES

A single controller couldn't solve many of the problems that a multi-controller approach has. In multi-controller systems, scalability is a major concern. So the researchers have introduced an SDN network with several controllers. Scalability remains an issue, however, because picking controller sites and assigning switches for various controllers are all part of the same process. Consistency, stability, and load balancing are all hampered by the presence of several controllers on the network. Even though SDN domains are controlled by different controllers, uniform network views must be maintained. It is necessary to safeguard the multi-uniformity controllers. Indeterminate failure and indeterminate attack on a range of controllers can influence the dependability of the control plane, independent of their kind or location on the control plane. For multi-controller research to be really effective,

it must identify ways to distribute the workload among several controllers such that no one controller suffers.

Controller Placement Problem (CPP): To begin, Heller et al. have improved the scalability of multi-controller CPP systems. There are two main concerns addressed by the CPP: how many controllers are required, and where they should be housed. In testing, control plane propagation latency is studied using two operational Internet installations and one hundred publicly available WAN topologies. Current reaction-time requirements may often be met with just one controller site, according to research that shows a link between latency and topology (though certainly not fault tolerance requirements). This study lacked an algorithm and theoretical demonstration, which was a pity.

Optimal Controller Placement: The authors provide a distributed technique based on non-

zero-sum games in order to deploy the multi-controller optimally. Saving money and improving service quality are the goals of this non-zero-sum game in which reward functions are generated for each controller and compared to the payoff values of the other controllers (QoS).

Bargaining Game: The placement of multicontrollers is also studied in the context of a game model. Data travel time between controllers and switches, data travel time between controllers, and controller processing time are all taken into account in this model. Multiple metrics are also taken into consideration in this model. One goal is to reduce communication latency, and another is to reduce communication overhead, and this study uses measurements to define an optimization issue. The authors play a negotiation game in attempt to find the optimal balance between the two objectives.

Mathematical Model: Using a mathematical model, the authors show how to find the ideal number of controllers, their position, and their kind in SDN. Various limitations are taken into account while trying to reduce the network's controller placement cost (e.g., controller capacity, path latency). Simulations show that this paradigm works well for planning small-scale SDN networks. Additionally, this model may be used to begin integrating SDN or plan a new SDN in various organizations and cloud-based networks. As a result, when applied to large-scale control topology designs, this paradigm becomes overly resource-intensive.

Control Plane with Hybrid Hierarchy: Several orders of magnitude less computing complexity may be achieved by using Orion, a hierarchical hybrid control plane, in large-scale SDN networks. Area controllers manage the physical switches and gather link information, whereas domain controllers monitor and manage the area controller devices as a layer beneath them. In contrast to Kandoo [25], the authors create an abstracted hierarchical routing approach between the area controller and domain controller layers [56-66].

CONCLUSIONS AND FUTURE DIRECTIONS

We conducted a systematic literature review on the topic of quality of service (QoS) in large-scale multi-controller software-defined networks (SDN) and examined various points of view in depth. An in-depth review and extraction of data from 66 research publications has been completed and is now available. The following were used to collect the result: (1) The architecture of SDN, (2) SDN controllers and evolution, (3) Relationship between SDN and QoS and (4) prospective future study topics. With regard to Quality of Service (QoS) in Multi Controller's Large Size Software Defined Networks (SDN), in recent years, Software Defined Networks (SDN) have made significant progress (SDN). Furthermore, opportunities and challenges are also discussed.

REFERENCES

- Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turetli, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3), 1617-1634.
- Singh, A. K., & Srivastava, S. (2018). A survey and classification of controller placement problem in SDN. *International Journal of Network Management*, 28, 1-25.
- A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), March 2010.
- N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69-74, 2008.

- T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, 2009, pp. 335-348.
- B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure," in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. HotNets-XI, New York, NY, USA: ACM, 2012, pp. 43-48.
- A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Intelligent design enables architectural evolution," in Proceedings of the 10th ACM Workshop on Hot Topics in Networks, ser. HotNets-X, New York, NY, USA: ACM, 2011, pp. 3:1-3:6.
- B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure," in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. HotNets-XI, New York, NY, USA: ACM, 2012, pp. 43-48.
- FEAMSTER N, REXFORD J, ZEGURA E. "The road to SDN: an intellectual history of programmable networks," ACM Sigcomm Computer Communication Review, vol. 44, no. 2, pp. 87-98.
- NADEAU T D, GRAY K. "SDN: Software Defined Networks," O'Reilly Media, Inc, 2013.
- Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). Multi-controller based software-defined networking: A survey. *IEEE Access*, 6, 15980-15996.
- Diego, F. M. V., Rothenberg, C., & Azodolmolky, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14-76.
- 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.
- 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).
- 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.
- Cai, Z., Cox, A. L., & Ng, T. S. E. (2011). Maestro: A system for scalable OpenFlow control. Rice University, Tech. Rep.
- Krishnaswamy, U., Berde, P., Hart, J., Kobayashi, M., Radoslavov, P., Lindberg, T., Sverdlow, R., Zhang, S., Snow, W., & Parulkar, G. (2013). ONOS: An open source distributed SDN OS. <http://www.slideshare.net/umeshkrishnaswamy/open-networking-system>.
- Saikia, D. (2013). MuL OpenFlow controller. <http://sourceforge.net/projects/mul/>.
- Nippon Telegraph and Telephone Corporation. (2012). Ryu Network Operating System. <http://osrg.github.com/ryu/>.
- McCauley, M. (2012). POX. <http://www.noxxepo.org/>.
- Juniper Networks. (2013). Opencontrail. <http://opencontrail.org/>.
- OpenDaylight. (2013). OpenDaylight: A Linux Foundation Collaborative Project. <http://www.opendaylight.org>.
- The Real-Time Cloud. (2014). <http://www.ericsson.com/res/docs/whitepapers/wp-sdn-and-cloud.pdf>.
- 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.*

- HP VAN SDN Controller Software.
<http://h20195.www2.hp.com/v2/getpdf.aspx/4AA4-9827E NW.pdf>.
- Floodlight is a Java-based OpenFlow controller. (2012). <http://floodlight.openflowhub.org/>.
- 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.
<https://www.scribd.com/document/117471434/OpenFlow-1-2>
- D. Dotan and R. Y. Pinter, "HyperFlow: an integrated visual query and dataflow language for end-user information analysis," 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), 2005, pp. 27-34.
- R. Y. Shtykh and T. Suzuki, "Distributed Data Stream Processing with Onix," 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, Sydney, NSW, 2014, pp. 267-268.
- A. M. Al-Sadi, A. Al-Sherbaz, J. Xue and S. Turner, "Routing algorithm optimization for software defined network WAN," 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA), Baghdad, 2016, pp. 1-6.
- M. Hungyo and M. Pandey, "SDN based implementation of publish/subscribe paradigm using OpenFlow multicast," IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, 2016, pp. 1-6.
- S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," 1st Workshop HotSDN, 2012, pp. 19-24.
- Karakus, M., & Duresi, A. (2017). Quality of service (QoS) in software defined networking (SDN): A survey. *Journal of Network and Computer Applications*, 80, 200-218.
- Oztemel, E., & Gursev, S. (2020). Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1), 127-182.
- Amin, R., Reisslein, M., & Shah, N. (2018). Hybrid SDN networks: A survey of existing approaches. *IEEE Communications Surveys & Tutorials*, 20(4), 3259-3306.
- Poggi, M., Tosi, F., & Mattocchia, S. (2018, September). Learning monocular depth estimation with unsupervised trinocular assumptions. In *2018 International conference on 3d vision (3DV)* (pp. 324-333). IEEE.
- Optimal scheduling for multi-flow update in software-defined networks, *Journal of Network and Computer Applications* 54 (2015) 11 { 19. doi:<http://dx.doi.org/10.1016/j.jnca.2015.04.009>.
- A multi-rat bandwidth aggregation mechanism with software-defined networking, *Journal of Network and Computer Applications* (2015) {doi:<http://dx.doi.org/10.1016/j.jnca.2015.11.003>.
- R. Braden, D. Clark, S. Shenker, RFC 1633 : Integrated Services in the Internet Architecture: an Overview (1994). URL www.ietf.org/rfc/rfc1633.txt
- L. Zhang, S. Berson, S. Herzog, S. Jamin, RFC 2205 : Resource ReSerVation Protocol (RSVP) { Version 1 Functional Specification (1997). URL www.ietf.org/rfc/rfc2205.txt
- 41.S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, RFC 2475 : An Architecture for Differentiated Service (1998). URL www.ietf.org/rfc/rfc2475.txt
- E. Rosen, A. Viswanathan, R. Callon, RFC 3031 : Multi-protocol Label Switching Architecture (2001). URL www.ietf.org/rfc/rfc3031.txt
- Software-Defined Networking: The New Norm for Networks, Tech. rep., Open Networking Foundation (ONF) (April 2012).

- 44.W. Li, W. Meng, L. F. Kwok, A survey on open ow- based software de ned networks:Security chal- lenges and countermeasures, Journal of Network and Computer Applications 68 (2016) 126 { 139.doi:http://dx.doi.org/10.1016/j.jnca.2016.04.011. URL <http://www.sciencedirect.com/science/article/pii/S1084804516300613>
- Secure and dependable software de ned networks, Journal of Network and Computer Applications (2015) {doi:http://dx.doi.org/10.1016/j.jnca.2015.11.012.
- R. Masoudi, A. Gha ari, Software de ned networks: A survey, Journal of Network and Com- puter Applications 67 (2016) 1 { 25. doi:http://dx.doi.org/10.1016/j.jnca.2016.03.016. URL <http://www.sciencedirect.com/science/article/pii/S1084804516300297>
- S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake,
- Finnegan, N. Viljoen, M. Miller, N. Rao, Are we ready for sdn? implementation challenges for software-de ned net-works, Communications Magazine, IEEE 51 (7) (2013) 36{ 43. doi:10.1109/MCOM.2013.6553676.
- 48.K. Bakshi, Considerations for software de ned networking (sdn): Approaches and use cases, in: Aerospace Conference, 2013 IEEE, 2013, pp. 1{9. doi:10.1109/AERO.2013.6496914.
- S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle,
- S. Stuart, A. Vahdat, B4: Experience with a globally-deployed software de ned wan, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIG-COMM '13, ACM, New York, NY, USA, 2013, pp. 3{14. doi:10.1145/2486001.2486019. URL <http://doi.acm.org/10.1145/2486001.2486019>
- SDN architecture, Tech. rep., Open Networking Foundation (ONF) (June 2014). URL https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf
- N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Open ow: Enabling innovation in campus networks, SIGCOMM Com-put. Commun. Rev. 38 (2) (2008) 69{74.
- S. J. Vaughan-Nichols, Open ow: The next generation of the network?, IEEE Computer 44 (8) (2011) 13{15. URL <http://dblp.uni-trier.de/db/journals/computer/computer44.html#Vaughan-Nichols11>
- P. Lin, J. Bi, S. Wol , Y. Wang, A. Xu, Z. Chen, H. Hu,Y. Lin, A west-east bridge based sdn inter-domain testbed, Communications Magazine, IEEE 53 (2) (2015) 190{197. doi:10.1109/MCOM.2015.7045408.
- OpenFlow Switch Speci cation, 1.5.0 (December 2014). URL <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>
- M. Fernandez, Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive, in: Advanced Infor-mation Networking and Applications (AINA), 2013 IEEE 27th International Conference on, 2013, pp. 1009{1016. doi:10.1109/AINA.2013.113.

- S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," 1st Workshop HotSDN, 2012, pp. 19-24.
- B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," 1st Workshop HotSDN, 2012, pp. 7-12.
- H. K. Rath, V. Revoori, S. M. Nadaf and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, NSW, 2014, pp. 1-6.
- A. Ksentini, M. Bagaa, T. Taleb and I. Balasingham, "On using bargaining game for Optimal Placement of SDN controllers," 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, 2016, pp. 1-6.
- A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks," in IEEE Communications Letters, 2015, vol. 19, no. 1, pp. 30-33.
- Y. Fu et al., "A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale Software-Defined Networks," in IEEE Transactions on Network and Service Management, 2015, vol. 12, no. 2, pp. 117-131.
- F. Yeung, "Internet 2: scaling up the backbone for R&D," IEEE Internet Computing, 1997, vol. 1, no. 2, pp. 36-37.
- B. Soper and J. Musacchio, "A non-zero-sum, sequential detection game," 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, 2015, pp. 361-371.
- D. Marconett, S. J. B. Yoo, Flowbroker: Market-driven multi-domain sdn with heterogeneous brokers, in: Optical Fiber Communications Conference and Exhibition (OFC), 2015, 2015, pp. 1-3. doi:10.1364/OFC.2015.Th2A.36.
- J. Wang, Y. Wang, X. Dai, B. Benasou, Sdn-based multiclass qos guarantee in inter-data center communications, Cloud Computing, IEEE Transactions on PP (99) (2015) 1-1. doi:10.1109/TCC.2015.2491930.
- J. Wang, Y. Wang, X. Dai, B. Bensaou, Sdn-based multiclass qos-guaranteed inter-data center trac management, in: Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, 2014, pp. 401-406. doi:10.1109/CloudNet.2014.6969028.