Proc. of the 8th International Conference on Engineering and Emerging Technologies (ICEET) 27-28 October 2022, Kuala Lumpur, Malaysia

Building a QoS Testing Framework for Simulating Real-World Network Topologies in a Software-defined Networking Environment

Hyeong Seon Yoo
Department of Information Systems and Computer Science
Ateneo de Manila University
Quezon City, Philippines
sylvesteryoo@naver.com

William Emmanuel S. Yu
Department of Information Systems and Computer Science
Ateneo de Manila University
Quezon City, Philippines
wyu@ateneo.edu

***Abstract*—Software-defined networking is an emerging technology for implementing Quality of Service (QoS) for its logically centralized control and decoupled control and forwarding planes. In this paper, we extend a previous framework that simulated a fat-tree topology software defined network under QoS provisioning techniques to support different topologies, including data from the Internet Topology Zoo. We compare round-trip times between each server and client, and transfer rates of HTTP traffic in the tree, mesh, and topology provided by Kreonet and Uni-C under a distributed class-based queuing system. We observe that while round-trip times are not significantly different, transfer rate of HTTP traffic as generated by ApacheBenchmark in Kreonet decreased 57.1109% from the fat-tree topology, and 49.5328% from the mesh topology. Uni-C topology took 114.9% more time for the switches in the network to have installed all their flows than the Kreonet topology, the next slowest topology. Uni-C topology also failed to finish the HTTP benchmarks due to its more complex topology in a resource-limited machine. These differences demonstrate that the complexity of real world network environments (i.e. route loops, layering, convergence times) affect the network behavior, which is key to a good simulator. Therefore, the modified framework is able to support and properly simulate real-world network topologies.**

***Index Terms*—network modelling, software-defined networks, QoS, network topology, Internet Topology Zoo**

I. Introduction

Quality of Service (QoS) is a technology that guarantees a certain level of quality in the provision of certain types of usually resource-intensive traffic [1]. It is increasingly more in-demand in increasingly complex and large networks. As such, it is important to develop and test QoS techniques, behaviors, and algorithms with real-world network data, and for simulation frameworks for QoS to be able to correctly simulate such networks. Requirements of QoS are enforced through queuing, which is the ordering of packets in traffic flow, and bandwidth control, which is how much of the network resources the packets take up [2]. In traditional networks with disparate forwarding devices with different configurations, queuing and bandwidth control is enforced through service-level agreements between the user and the network provider [2]. This does not allow for an abstract and fine-grained control of network traffic.

Software-defined networking (SDN) is an approach to networking that solves the problem of fragmented network management and configuration by decoupling the data plane from the control plane. Forwarding devices in the network are logically centralized, and uses a protocol such as OpenFlow to control behavior. Software-defined networking allows for QoS provisioning to be implemented with a more abstract view of the network, allowing for better control of different types of traffic. Advantages of the SDN paradigm in providing QoS include custom routing of QoS traffic instead of shortest-path routing, better QoE through additional parameters to the traditional QoS metrics, and better monitoring of network dynamics [2].

Researchers have tested many QoS algorithms in software-defined networks on Mininet with a network operating system such as Ryu and OpenFloodLight. Regencia and Yu [3] built a testing framework with Mininet and Ryu to test various QoS algorithms under different network topologies. However, the study did not investigate the performance of these QoS algorithms in different topologies. The algorithms can be more tested with real-world data from the Internet Topology Zoo, which provides data from network operators about live networks that provide Internet service [4]. In this paper, we present a modification of the framework to use Internet Topology Zoo data described with GraphML, as well as some model, typical network topologies, to test QoS queueing algorithms. We present a comparison of network convergence times, ping times, and ApacheBenchmark tests as synthetic traffic to the implemented network topologies under a basic class-based queuing system.

II. Related Literature

A. QoS and Software-defined Networking

The goal of QoS is to provide a certain level of guarantee to resource intensive or performance sensitive traffic flow from important applications such as VoIP, video conferencing,

and online gaming. These services are provided with certain guarantees in network performance metrics such as delay, bandwidth, jitter, and packet loss[5]. However, traditional methods of QoS provisioning suffer from scalability issues or a lack of fine-grained control [2]. Manual configuration of disparate devices is not suitable to meet the constantly changing demands of QoS. Software-defined networking alleviates this problem by providing a logically central point of control that can abstract the network into a consistent API that network applications can work with [6]. Because of this, QoS Provisioning with Software-defined networking has been an important topic of research in the last decade, with many attempts to simulate various conditions and QoS provisioning methods over different platforms and environments.

B. QoS and Network Topology

Different network topologies have different applications, strengths and weaknesses. Mesh networks, for example, are used to maintain reliable and self-healing connections throughout the network [7]. Ring networks are used mostly in local area networks where connection is maintained even if one node fails, and since each node has exactly two neighbors, routing is simple. The tree topology is suitable for large geographical areas and thus can be implemented for large-scale networks such as Metropolitan Area Networks [8]. As these networks have different characteristics and applications, some QoS research has been done with different topologies. For example, Laassiri, Moughit, and Idboufker [9] evaluates packet loss, gigue, latency, and end-to-end delay in star, ring, tree, and hierarchical topologies in an SDN network. Kaiming Liu et. al. [10] introduced an algorithm that routes packets according to QoS constraints in a wireless mesh network using heuristics based on historical search results with the Dijkstra's algorithm.

C. Testing various QoS techniques in SDN

Community-driven open-source Network Operating System (NOS) software such as Floodlight, NOX, POX, and Ryu have been used in research for testing and simulation purposes. These NOS softwares, in conjunction with Mininet, are used as a testbed for prototyping and testing various network algorithms, including QoS provisioning techniques. Reference [11] used Mininet to test 5 different scenarios for the FlowBroker QoS technique, while [12] used 5 switches, 2 servers and 11 clients to test the HiQoS technique. Regencia and Yu [3] presented a Mininet and Ryu based framework for simulating various QoS queue management algorithms based on a previous work by Chato and Yu [13] which tested various Class-based queuing (CBQ) algorithms. The framework accepts custom QoS algorithms and custom parameters for a tree topology as input to a Ryu controller that implements a Level 2 learning switch with the custom algorithm.

However, it is important to test these algorithms in real-life situations. We can approximate live networks in Mininet using the Internet Topology Zoo, which was used by Mamushiane, Mwangama, and Lysko [14] to test solutions to the placement of controllers in an SDN topology . Metter [15] also investigated the impact of network topology on the processing times of SDN controllers with 261 topologies from the Internet Topology Zoo. Given that we have topology data, and that the Regencia and Yu framework was intended to test different QoS algorithms in many situations [3], modifying it to include these topology data will allow for easier testing of simple and complex QoS techniques in different situations.

III. Framework and Methodology

A. Building the simulation framework

First, we expand the QoS Testing Framework (testing framework) from Regencia and Yu to support different classes of topologies: fat-tree topologies with fanout greater than 2, the complete mesh, and general graphs represented by GraphML data. To accomplish this, we modify how the *topology configuration* file is used in the testing framework. In the original testing framework, the topology configuration file contains information about the number of clients and the number of layers in a tree topology, which is then read by *configuration generator* scripts to reconstruct the topology within itself to generate the correct configurations.

We introduce an intermediary script called the *Topology Information Generator* contains the implementation of various topologies. The Topology Information Generator then can emit a *Topology Information* file that contains all the nodes and links of the network can be read by the configuration generators to generate the necessary configuration files. For standard topologies such as the fat-tree and complete mesh, the topology information generator directly includes the implementation details. For data from the Internet Topology Zoo, the topology information generator parses a GraphML file with the NetworkX library to emit the same topology information file.

The framework is further modified so that the Ryu Controller script uses the STP protocol to support topologies that contain loops and multiple paths to a certain node in the network, which is commonly found in live networks for resiliency in the network. The reference simple_switch_stp_13.py code from the Ryu Library is used as a basis to develop the controller that will run on any topology and with any QoS queue management technique.

B. Configuration of the tested network topologies

To implement different topologies within the testing framework, we take the tree topology from the original study by Regencia and Yu [3] and modify the number, arrangement, and links in the switches that connect the "core" switch to the clients. The "core" switch is the switch that connects to a switch connecting all servers in a star topology formation. In Fig. 1, the topology to be changed corresponds to the circled switches.

The "core" switch in the fat-tree topology remains to be the root of the tree. In the complete mesh topology, the switch with the numerical label 1 is considered to be the root

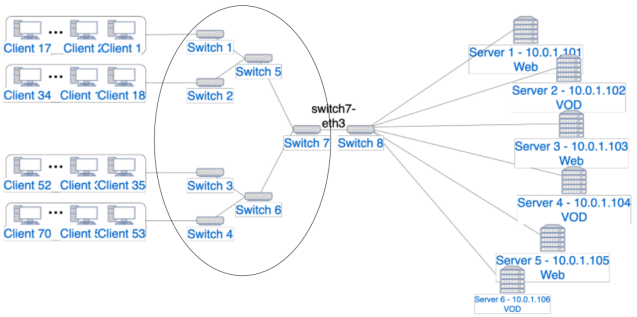

Fig. 1. Diagram of a fat-tree topology tested in Regencia and Yu

TABLE I
SET-UP SPECIFICATION USED FOR RUNNING THE SIMULATIONS

Processor	AMD Ryzen 5 4500U at 2.3 GHz, 3 out of 6 cores
RAM	2.3 GB,
Virtual Machine Software	QEMU-KVM
Mininet version	2.3.0
Ryu version	4.30

switch, since a mesh topology has at least one symmetry with respect to any node in the mesh. For the topologies from the Internet Topology Zoo, we simulate the worst case scenario by assigning the core as the node with the highest number of total hops from all other nodes, which can be calculated with the Floyd-Warshall algorithm.

Clients are connected to all other switches, thereby simulating the worst-case scenario where client traffic comes from all parts of the topology. However, for the tree topology, only the leaf nodes will be used to connect the clients. The clients will be distributed equally among the switches to be used for client connection. Finally, we connect four servers to the server switch to complete the topology. The diagrams of the tested topologies are shown in diagrams a, b and c.

We configured the following topologies for simulation:

1) Tree topology with two layers and fan-out 3
2) Complete Mesh topology with 5 vertices in the mesh
3) Kreonet network data provided by the Internet Topology Zoo
4) Uni-C network data provided by the Internet Topology Zoo

Figures 2, 3, 4, and 5 show the topologies tested in this paper. In all figures, red triangles represent servers, green triangles represent clients, and yellow squares represent Openflow-enabled switches. Numbers in each shape represent the label of the client, server or switch. We also attempted to test the Uni-C network data, but the simulation did not complete properly, as we show in Results and Analysis.

The networks are set-up in the environment described in Table I for simulation.

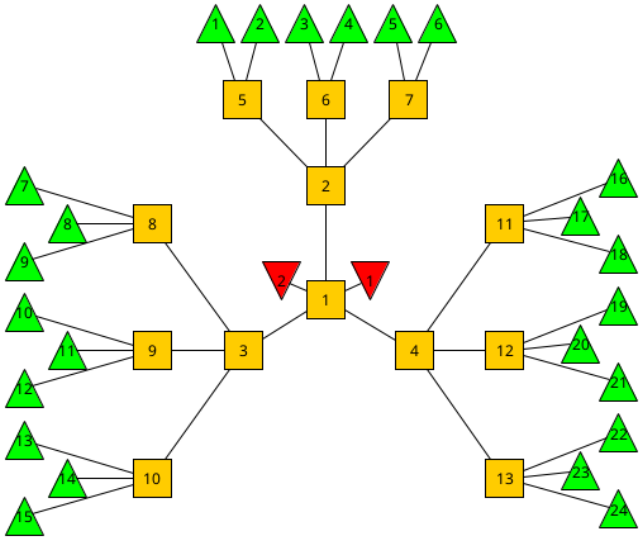

Fig. 2. Tree topology with fanout 3 as configured in the simulation.

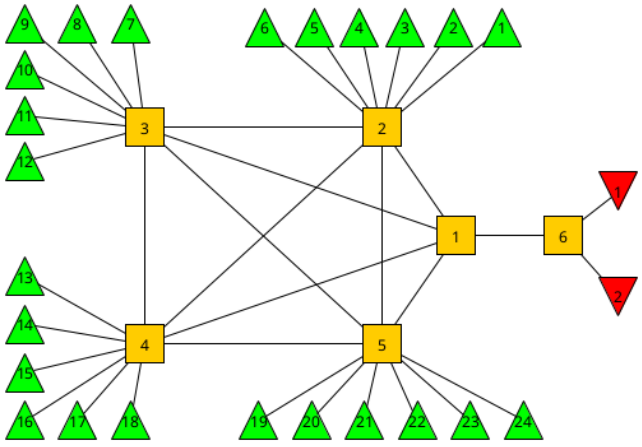

Fig. 3. Complete Mesh Topology with 5 vertices in the mesh.

C. Server Configuration

We set up 2 servers, each running a Python server from the http.server library, hosting two files as shown in Table II.

D. Client Configuration

We tested 24 clients for every topology. Every client ran an ApacheBenchmark test simultaneously. Each client requested for one file in the server, and every file in each server

TABLE II
FILES HOSTED BY THE TWO PYTHON HTTP.SERVER SERVERS

Server ID	Server IP Address	File Size and Type	
		Low Load	High Load
Server 1	10.0.1.101	100 kB JPEG	10 MB JPEG
Server 2	10.0.1.103	16 MB PDF	100 MB PDF

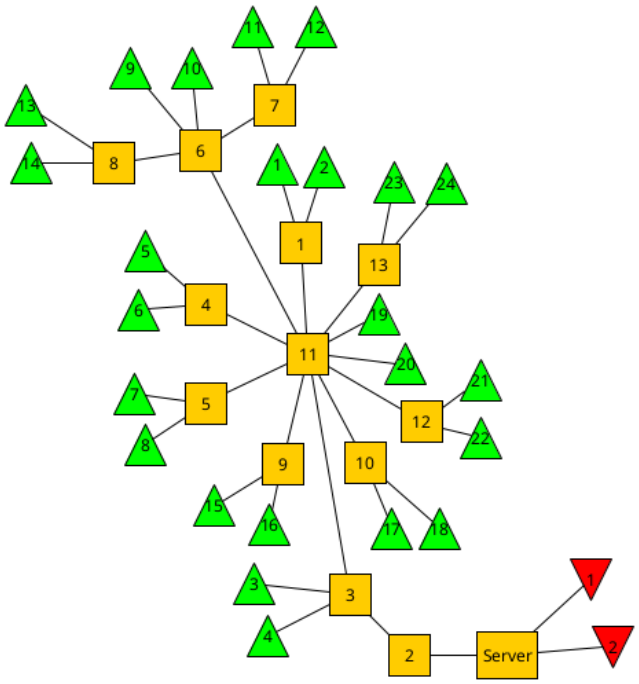

Fig. 4. Kreonet network data topology.

TABLE III
CONFIGURATIONS FOR CLIENTS DENOTING THE SERVER AND THE LOAD THAT THE CLIENT WILL REQUEST IN THE SIMULATIONS

Client numbers	Server	Load
1, 5, 9, 13, 17, 21	Server 1	Low
2, 6, 10, 14, 18, 22	Server 2	Low
3, 7, 11, 15, 19, 23	Server 1	High
4, 8, 12, 16, 20, 24	Server 2	High

will get requests as equally as possible. Table III shows the configuration of these clients.

E. Quality of Service configuration

The parameters for QoS Ports are identical to the configuration in Regencia and Yu [3]. This study uses the leaf-enforced basic class-based queuing QoS algorithm. Table IV shows the queue configuration applied to each port of the leaf used for this study. In this study, all pings will be directed to queue 2, and HTTP requests will be sent to queue 0. Queue 1 will be unused as the experiment does not test UDP data.

TABLE IV
CONFIGURATIONS USED FOR PORTS WHERE QOS PARAMETERS ARE ENFORCED

Queue	Type of traffic	Minimum and Maximum bandwidth	Priority
Q_0	TCP	333.33 Mbit/s	0
Q_1	UDP	333.33 Mbit/s	1
Q_2	Others(ICMP, ARP)	333.33 Mbit/s	2

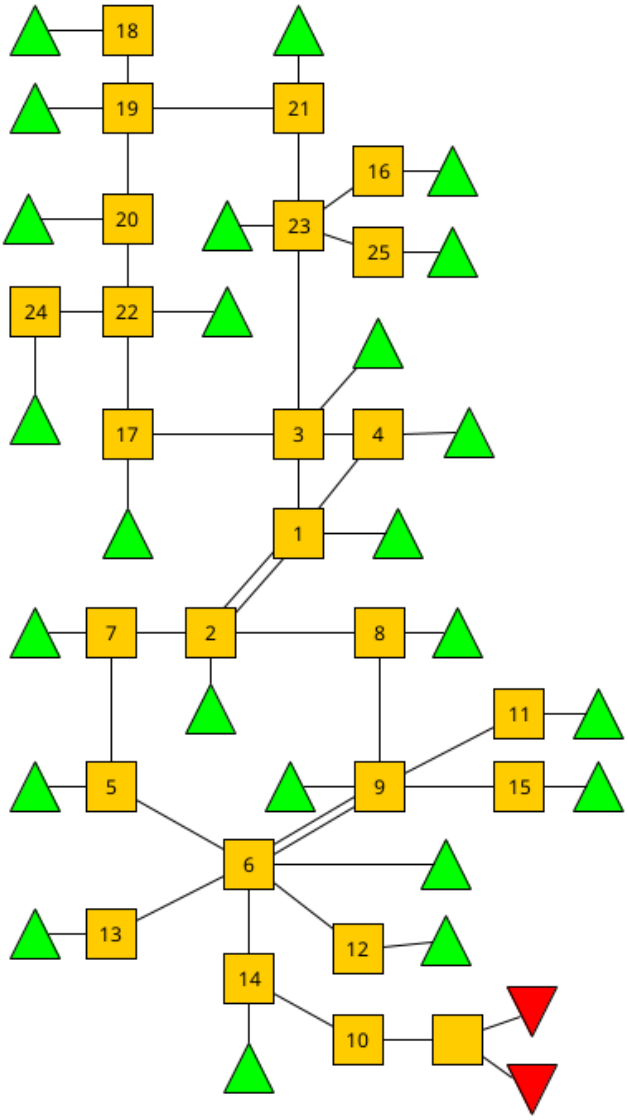

Fig. 5. Uni-C network data topology.

F. Running the Simulation

Fig. 6 shows the procedure that was used to run the simulations. We note that initially, the hosts cannot reach each other since the STP protocol takes time to converge. For this reason, we wait for 40 seconds before running the benchmark tests. We also note that initially, the OpenFlow-enabled switches do not have flows installed in them, therefore, we ping Server 1 from Client 1 to make sure all the flows are installed. This works due to ARP requests and responses being flooded to all switches, which allow the Ryu learning switch application to fill their MAC Address tables and install the flows into the switches. We measured the sum of the round-trip times before they are consistently below 1 second, which we call the "convergence time" of the network.

After the simulation system has been set-up, and the net-

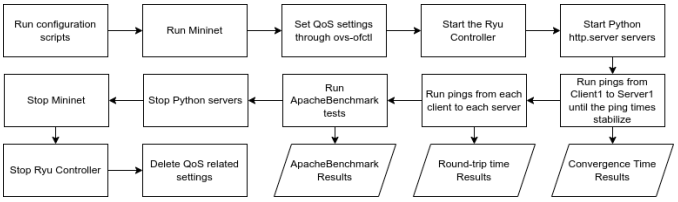

Fig. 6. Flowchart representing the procedure used for simulation tests

TABLE V
AVERAGE PING TIME FROM ALL CLIENTS TO ALL SERVERS, IN MILLISECONDS

Topology	Fat-tree		Mesh		Kreonet	
Server	1	2	1	2	1	2
Average	0.183	0.181	0.178	0.181	0.169	0.175
SD	0.00883	0.00861	0.00966	0.011	0.0144	0.0122

work has "converged", we exeecuted ping commands in each client to all servers 20 times, and we took the average of the last 10 pings, as the initial ping may need instructions from the controller due to the lack of flows. In fact, after the network has converged, each subsequent ping test took only one or two pings to show consistent results.

Once the round-trip times have been collected between each server and client, we conducted the ApacheBenchmark test for all clients as configured in Table III simultaneously for 4 minutes.

IV. RESULTS AND ANALYSIS

We first note that because of the limited capacity of the testing environment, in the Uni-C topology simulation which has 25 switches, some clients in the network did not reliably complete the ApacheBenchmark test. This is expected behavior because the virtual machine running the Mininet platform is limited in capacity, therefore topologies with many devices will run slowly. Therefore, we only present the convergence time for the Uni-C topology, which is much higher than the rest of the tested topologies. For the rest of the tested topologies, we also present round-trip time results and ApacheBenchmark results from simulations.

Table V shows the average round-trip times from each client to the two servers for each topology. All topologies show similar ping times. Table VIII shows the convergence times of the network topologies, and we find that while the fat-tree topology and the Kreonet topology takes a similar amount of time, the Mesh topology takes little time for the ping times to drop under 1 second. This may be due to the complexity of the topology, as the Mesh topology tested has 5 nodes only, while both the fat-tree topology and the Kreonet data has 13, which is close to expected behavior. Meanwhile, Uni-C data shows a much higher convergence time for the network, a 114.9% increase from the next highest Kreonet network. This is also expected since the Uni-C topology is more complex, having nearly twice as many switches in the network than the Kreonet topology.

TABLE VI
AVERAGE TOTAL TRANSFER FOR EACH TYPE OF REQUEST

Server / Load Type	Fat-tree (bytes)	Mesh (bytes)	Kreonet (bytes)
Server 1 / Medium	139,153,418	340,311,521	175,412,517
Server 1 / High	5,321,316,224	2,993,196,013	971,012,383
Server 2 / Medium	3,048,428,693	3,853,660,507	1,999,523,059
Server 2 / High	3,060,832,900	2,783,192,884	2,395,197,626
Across all clients	2,892,432,809	2,492,590,231	1,385,286,396

TABLE VII
AVERAGE TRANSFER RATE FOR EACH TYPE OF REQUEST

Server / Load Type	Fat-tree (kB/s)	Mesh (kB/s)	Kreonet (kB/s)
Server 1 / Medium	565.94	1384.53	713.34
Server 1 / High	21616.47	12142.55	2608.42
Server 3 / Medium	12307.32	14968.78	8121.14
Server 3 / High	12178.99	11165.19	8572.89
Across all clients	11667.18	9915.26	5003.95

Regarding HTTP load performance, Table VII shows that different sizes of HTTP load had different performance characteristics. In medium sized file types, the mesh topology performed best, while in large files (high load), fat-tree topology performed the best, with Kreonet topology performing the slowest in all file types except for the smallest 100-kilobyte jpeg requests. On average, the transfer rate of Kreonet was the worst with a decrease of 57.1109% from the fat-tree topology and 49.5328% from the mesh topology. This result shows that there is indeed a difference in the performance of the different topologies under the simulations done in the previous study.

The results show that the simulator exhibits expected behavior as the complexity of the network (loops, topology depth, number of switches) increases in terms of convergence time and synthetic traffic performance. The more complex, real-world networks showed slower convergence times and slower synthetic traffic transfer rate. The result that the more complex Uni-C topology failed to finish all the ApacheBenchmark tests also shows that the simulator exhibits expected behavior. This points to more potential research that can be done with various QoS methodologies and network topologies using this framework.

V. CONCLUSION

We have successfully modified the testing framework to simulate some ideal topologies and real-life network topologies from the Internet Topology Zoo. We were able to compare the round-trip times and bandwidth of these topologies through synthetic benchmarks. We observed that there are differences

TABLE VIII
NUMBER OF PINGS TAKEN FOR THE TOPOLOGY TO CONVERGE (I.E. FOR ROUND-TRIP TIMES DROP TO BELOW 1 SECOND)

Metric	Fat-Tree	Mesh	Kreonet	Uni-C
Number of Pings	30	28	26	32
Total time of pings(ms)	626.2	198.5	806.3	1732.5

in the performance of ApacheBenchmark between network topologies under the tested QoS constraint and under the leaf-enforced QoS algorithm. The difference of behaviors between the fixed topologies (tree, mesh) and the live topologies (Kreonet, Uni-C) are as expected, which demonstrates that this framework can be used to simulate networks that more closely reflect real-world conditions.

This paper is a step towards future work which will include different types of traffic and different QoS algorithms for further analysis. Future work will also use more powerful hardware to allow for the simulation of larger topologies such as the Uni-C topology and other more complex topologies in the Internet Topology Zoo.

REFERENCES

[1] P. Ferguson and G. Huston, *Quality of service: delivering QoS on the Internet and in corporate networks*. New York: Wiley, 1998.

[2] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, Feb. 2017.

[3] J. E. T. Regencia and W. E. S. Yu, "Introducing a Test Framework for Quality of Service Mechanisms in the Context of Software-Defined Networking," in *Proceedings of Sixth International Congress on Information and Communication Technology*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds. Singapore: Springer Singapore, 2022, vol. 236, pp. 687–699, series Title: Lecture Notes in Networks and Systems.

[4] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[5] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. H. Schiller, "A concept for qos integration in web services." in *WISE Workshops*, 2003, pp. 149–155.

[6] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[7] M. Chawla, A. Mundra, N. Rakesh, A. Agrawal, and S. P. Ghrera, "Fault tolerance based routing approach for WMN," in *2015 International Conference on Computer and Computational Sciences (ICCCS)*, Jan. 2015, pp. 177–182.

[8] M. Gerla and L. Fratta, "Tree structured fiber optics MANs," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 6, pp. 934–943, Jul. 1988.

[9] F. Laassiri, M. Moughit, and N. Idboufker, "Evaluation of the QoS parameters in different SDN architecture using Omnet 4.6++," in *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. Monastir: IEEE, Dec. 2017, pp. 690–695.

[10] Kaiming Liu, Yahui Cao, Yuanan Liu, Gang Xie, and Chao Wu, "A novel min-cost Qos routing algorithm for SDN-based wireless mesh network," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. Chengdu, China: IEEE, Oct. 2016, pp. 1998–2003.

[11] D. Marconett and S. J. B. Yoo, "FlowBroker: A Software-Defined Network Controller Architecture for Multi-Domain Brokering and Reputation," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 328–359, Apr. 2015.

[12] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: An SDN-based multipath QoS solution," *China Communications*, vol. 12, no. 5, pp. 123–133, May 2015.

[13] O. Chato and W. E. S. Yu, "An Exploration of Various Quality of Service Mechanisms in an OpenFlow and Software Defined Networking Environment in Terms of Latency Performance," in *2016 International Conference on Information Science and Security (ICISS)*. Pattaya, Thailand: IEEE, Dec. 2016, pp. 1–7.

[14] L. Mamushiane, J. Mwangama, and A. A. Lysko, "Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?" in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Verona, Italy: IEEE, Nov. 2018, pp. 1–6.

[15] C. Metter, S. Gebert, S. Lange, T. Zinner, P. Tran-Gia, and M. Jarschel, "Investigating the impact of network topology on the processing times of SDN controllers," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, ON, Canada: IEEE, May 2015, pp. 1214–1219.