Performance Characteristics of Selected Network Topology in a Software-Defined Networking QoS Testing Framework

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

***Abstract*—Quality of Service in SDN/OpenFlow can be provided by making use of queues and their management techniques. This allows for better and more fine-grained control of network traffic. In this study, we test a Class-based Queuing (CBQ) algorithm for QoS provision that classifies network traffic according to the application (e.g., video traffic or HTTP traffic) over a selection of topology forms from the Internet Topology Zoo (ITZ). To accomplish this, we utilize a Mininet-based QoS Testing framework that allows for testing of various QoS algorithms. To select the networks to be tested, we use K-means clustering over the ITZ dataset according to network size for random sampling within each cluster. We show that under CBQ, HTTP stress test performance was better in clusters with smaller network size, while VLC video performance remained similar throughout clusters. KPI analysis also shows that clients closer to servers exhibited better performance than those farther away, while VLC video performance remained consistent throughout the network. The results show that the framework is able to satisfy QoS requirements while showing performance differences across various simulated network topologies.**

Keywords—software-defined networking, QoS, network topology, class-based queuing

I. Introduction

Nowadays, providing Quality of Service (QoS) guarantees for various types of networks and network traffic is increasingly in demand. In view of this, Software-defined networking is being investigated as a simpler, more centralized way to implement the demands of the QoS constraints. Networks today employ various topologies including various degrees of mesh networks in lieu of basic topology shapes such as the tree, ring, and star networks to improve reliability and resilience to failure [1]. For this reason, we investigate in this paper the behavior of Class-based Queuing (CBQ) algorithm for QoS queue management in selected topologies in the Internet Topology Zoo (ITZ) [2]. To accomplish this, we use a framework based on Mininet and Ryu presented in [3]. Which allows testing of various QoS Provisioning techniques in various topologies. The aforementioned framework is an improvement upon [4]. Which was limited to tree topologies. Reference [3] demonstrates expected behaviors across four different networks with different properties. In continuation of [3]. We simulate a sample of 10 semi-random topologies from the ITZ by grouping the networks in the ITZ with K-means clustering, and then obtaining a random sample from each of the clusters. We then employ the Key Performance Indicator (KPI) analysis to gain new insights on the method employed in the previous research.

II. Related Literature

A. *QoS Provisioning in SDN*

Traditional QoS provisioning has two major frameworks: IntServ and DiffServ [5]. On one hand, IntServ works on the principle that routers need to reserve network resources to provide QoS for specific traffic flows. This allows for a more fine-grained control of traffic, but it is difficult to manage many traffic flows [5]. DiffServ, on the other hand, classifies and marks individual packets to determine their per-hop behavior in the network [6]. Because of the inherent limitations of DiffServ and IntServ, many studies use OpenFlow/SDN as a mechanism to provide QoS demands. SDN is well-suited for QoS because it provides many features that enable better control of different traffic according to many different aspects, including but not limited to Multimedia Flow Routing, Resource Reservation, and Queue Management and Scheduling [7].

Researchers have studied various problems such as routing and queuing in the context of QoS with Software-defined networks. For example, Guck et. al. Studied different QoS routing algorithms based on shortest-path algorithms over a variety of topology shapes and sizes [8]. Yan et al. Proposed HiQoS, a combination of multipath routing and source IP based classification to better satisfy QoS constraints [9]. However, the testing of the algorithm was limited to one topology composed of 5 switches, 11 clients and 1 server. This limitation is important as it is crucial that the proposed methods are tested under different topology.

B. *Related Work on QoS Queueing Mechanisms*

An important feature of OpenFlow that enables traffic control based on traffic type is queue configuration, which is related to both multimedia routing and queue management mechanisms [7]. Many studies were conducted to make use of this feature. Sharma et al. implemented a QoS framework with three queues, and used traffic type to differentiate the traffic flow. They tested their framework on a pan-European network autonomous system, and a multiple-autonomous system topology developed in the CityFlow project [10]. In their testing, they measured the traffic speed of high-priority traffic and best-effort traffic, and simulated a link-failure during the simulation to demonstrate that their framework was able to maintain precedence of high-priority traffic even after failure conditions. Ishimori et al. proposed the use of multiple packet-scheduler queues to improve QoS, but only used a linear topology with two hosts to test their method [11]. They tested the response time of internal QoS operations for different queuing algorithms, as well as a Quality of Experience (QoE) metric named Peak Signal-to-Noise Ratio (PNSR) with First-in-first-out and Stochastic Fairness Queuing algorithms. Sudiharto et al. tested Priority Queuing versus CBQ in a VoIP context within the NS2 simulator [12]. They measured the delay of VoIP transmission over different bandwidths, and concluded that CBQ is more appropriate in their use case. However, like other research, their methodology was limited to a two nodes and two users.

Some previous studies have focused on the testing of various queuing techniques that are based on packet characteristics such as the application type, source and destination. Regencia and Yu introduced a framework that allowed the testing of various QoS queuing algorithms using Mininet, a popular SDN prototyping tool, and the Ryu network operating system [4]. They tested Class-based Queuing, Source-based Queuing, and Source/destination-based Queuing. However, the aforementioned study was limited to testing tree networks. The framework in [4] was extended by Yoo and Yu to support the testing of any network topology specified with GraphML files by using the built-in spanning-tree protocol (STP) libraries provided by the Ryu NOS [3]. They were able to demonstrate the different characteristics of a tree topology, a complete graph mesh topology, and two topologies selected from the ITZ.

While there are many studies that propose new queuing mechanisms accompanied by tests using their own metrics, there is a significant lack of comparative study of these mechanisms over different network topologies. This paper partially bridges this gap in literature by testing CBQ in a variety of selected topologies from the ITZ.

III. Theoretical Framework

A. *Selecting topology from the Internet Topology Zoo*

We survey only networks from the ITZ with at most 40 nodes, from networks whose geographic extent is labeled “Country” and “Region”. This limit is due to resource constraints of the testbed computer. We also remove networks which are previous versions of the same data provider. For example, BELNET provided topology data from 2003 to 2010. In this case, only the 2010 data was included in the set of networks that we sampled. This results in 196 topologies. We used the K-means clustering provided by `scikit-learn` with the number of nodes and the number of edges as the features [13]. This was done as an additional step to ensure that we sample a variety of topology with different characteristics. A plot of the four clusters formed from the 196 topologies is shown in Fig. 1. We randomly selected a total of 10 topologies, two from clusters 1 and 3 and three from clusters 0 and 2, in accordance with the size of the clusters. The characteristics of the selected topologies are summarized in Table I. We also included a tree topology with a fanout of 3 and a depth of 2, and a complete mesh network with 5 nodes as a baseline for comparison, as shown in Table II.

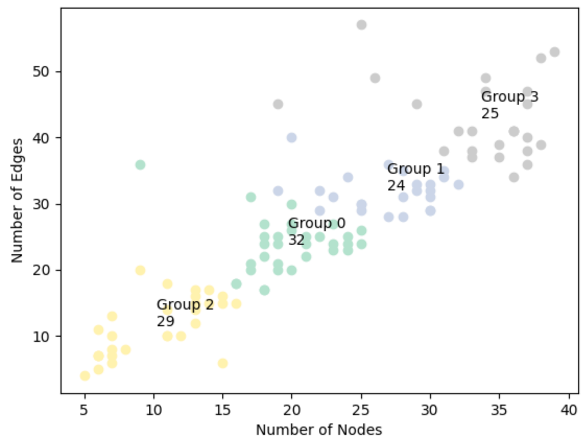

Fig. 1. Clusters given by the K-means clustering algorithm as implemented by `scikit-learn`.

B. *Topology Configuration*

We set up 48 clients which were connected to all nodes in the network except for one node designated as the “root” of the network. The “root” node is the node that has the largest sum of the number of hops to all other nodes in the network. If there are ties, then the node with a lower ID number as provided in the ITZ dataset is selected. The root node is connected to another switch, which connects to four server hosts in the network. The client hosts were connected to all nodes except for the root as evenly as possible, that is, the difference of the number of clients connected to any to given nodes is at most 1. As an exception for the fat-tree topology, clients were connected only to the leaves of the tree. Fig. 2 through Fig. 7 show some of the configured topologies. The 48 client hosts were assigned to test certain load types as shown in Table III for testing the four servers as configured in Table IV and V.

C. *QoS Configuration*

We use the CBQ algorithm implemented at the leaves as the QoS provisioning algorithm. We classify the network traffic relevant to the experiment according to Table VI. The queues are applied to the network with the `ovs-ofctl` command at the edge switches of the network, but not at the root node as described earlier.

D. *Key performance indicators per client*

We analyze the average data across all clients, but this only gives us a partial picture of the performance characteristics of each network. Therefore, we evaluate the key performance indicators (KPI) of each client in the network. The performance indicators we selected are the transfer rate for HTTP traffic and the average demux bitrate for video traffic because these are the relevant metrics for most network users.

TABLE I. SELECTED TOPOLOGY AND THEIR CHARACTERISTICS

Cluster	Name	Number of nodes	Number of edges	Qualitative description
0	Savvis	19	20	Two rings of size 11 and 8, but sharing a common edge
	Atmnet	21	22	Two big rings of size 12 and 9, but sharing a common edge
	Ibm	18	24	Sparse mesh network
1	Agis	25	30	4 connected rings with branches of depth 1
	WideJpn	30	33	Rings of size 6, with tree structure rooted on some nodes with fanout up to 6
2	Gridnet	9	20	Small, dense mesh network
	Nsfnet	13	15	Sparse mesh with three rings
	Singaren	11	10	A star topology with 9 branches
3	Janetbackbone	29	45	Sparse mesh network
	Canerie	32	41	Sparse mesh network

TABLE II. BASELINE TOPOLOGY AND THEIR CHARACTERISTICS

Name	Number of Nodes	Number of Edges	Qualitative Description
Fat Tree	13	12	A fat-tree with fanout 3 and 2 layers
Mesh	5	10	A network in the form of the complete graph K_5

TABLE III. SERVERS AND LOAD TYPES ASSIGNED TO EACH CLIENT

Client numbers	Server	Load
1, 5, 9, 13, … 41, 45	Server 1	Low
	Server 2	Low
2, 6, 10, 14, … 42, 46	Server 3	Low
	Server 4	Low
3, 7, 11, 15, … 43, 47	Server 1	High
	Server 2	High
4, 8,. 12, 16, …, 44, 48	Server 3	High
	Server 4	High

TABLE IV. 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 3	10.0.1.103	16 MB PDF	100 MB PDF

TABLE V. FILES HOSTED BY THE TWO VLC RTSP SERVERS

Server ID	Server IP Address	Video Resolution	
		Low Load	***High Load***
Server 2	10.0.1.102	360p	480p
Server 4	10.0.1.104	480p	720p

TABLE VI. NETWORK CLASSIFICATION AND THEIR ASSOCIATED PRIORITIES (HIGHER NUMBER MEANS HIGHER PRIORITY)

Application	Traffic Type	Bandwidth (MB/s)	Queue Priority
VLC Video Streaming	UDP and RTSP	333.33	2
PDF and JPG files	HTTP	333.33	1
Other traffic	ARP, ICMP	333.33	0

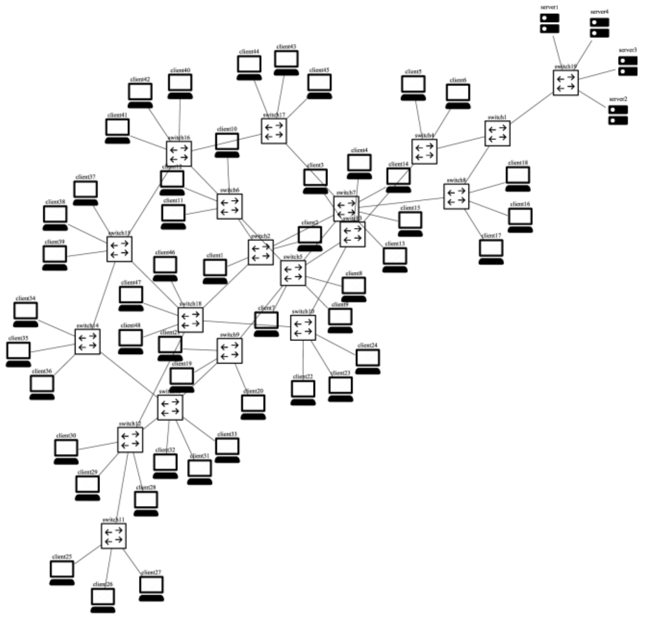

Fig. 2. Ibm Topology from the ITZ, representing Cluster 0

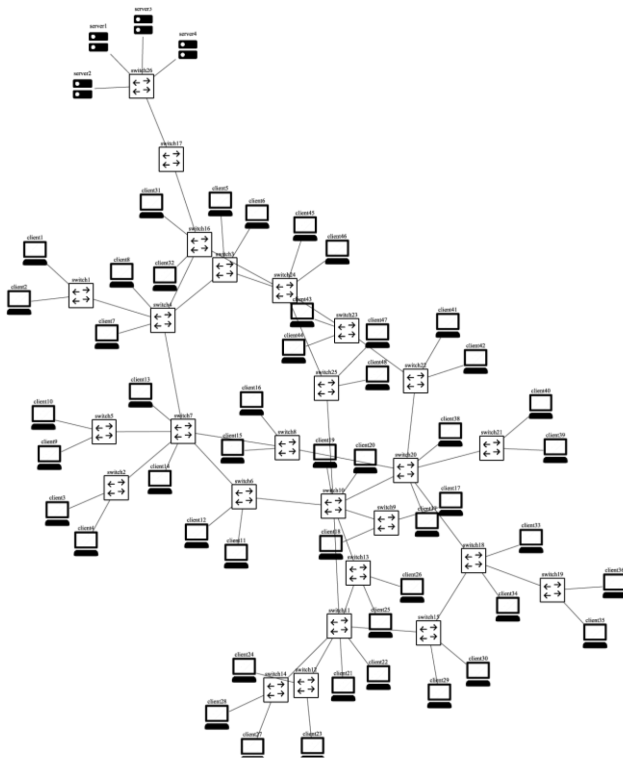

Fig. 3. Agis Topology from the ITZ, representing Cluster 1

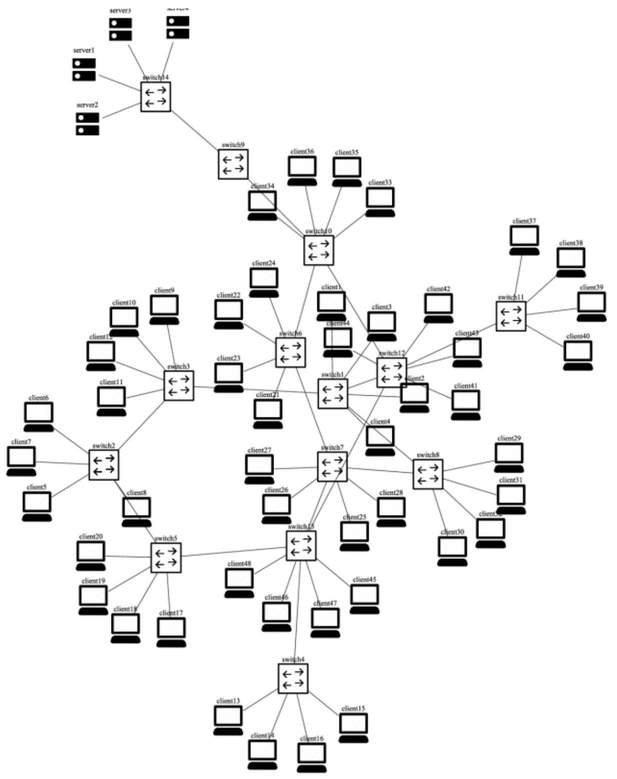

Fig. 4. Nsfnet topology from the ITZ, representing cluster 2

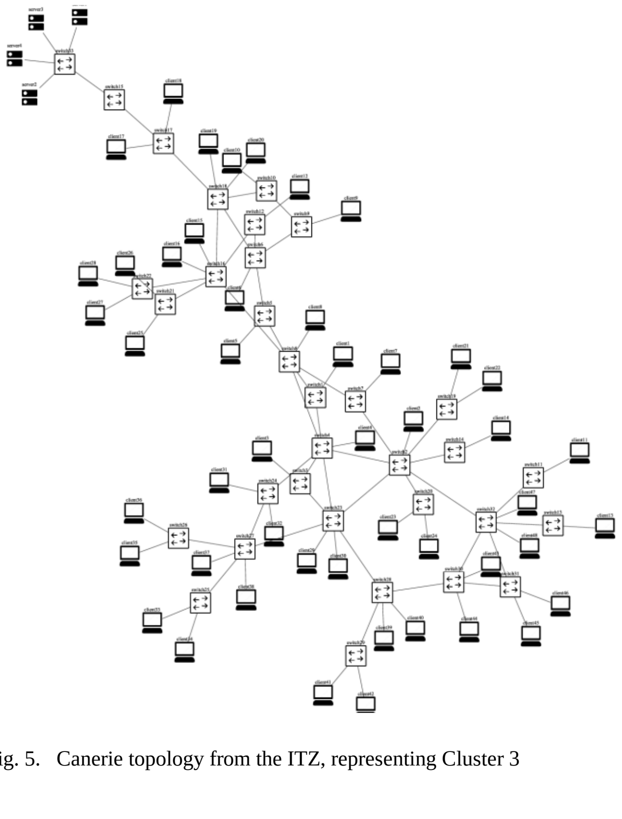

Fig. 5. Canerie topology from the ITZ, representing Cluster 3

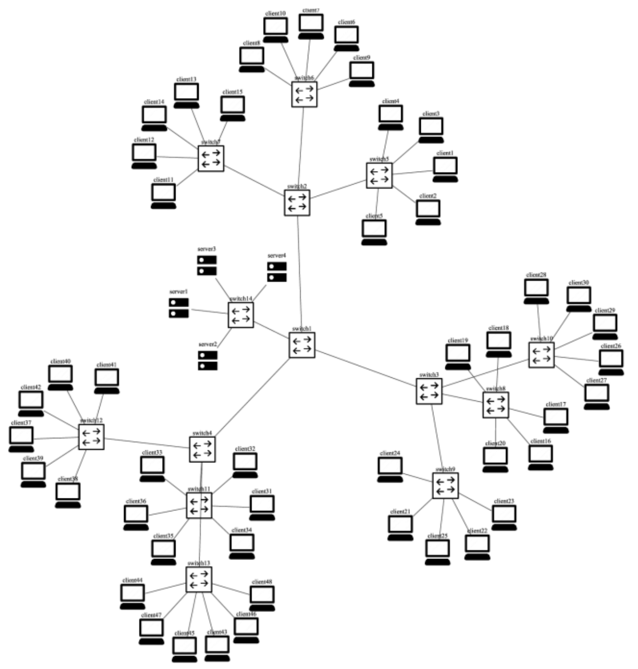

Fig. 6. Fat-tree topology with fanout 3 and depth 2

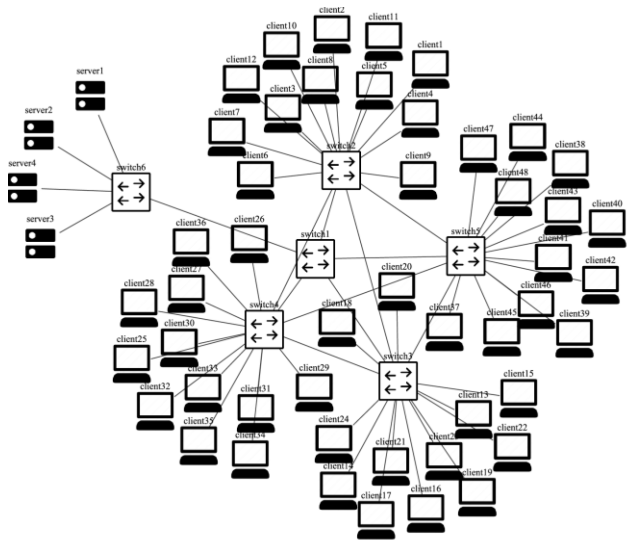

Fig. 7. 5-node mesh topology

IV. Methodology

In this paper, we modified the methodology described in [3] to benchmark the networks. Video traffic generation using VLC and RTSP remained the same. However, we replaced the ApacheBench program with the more modern and well maintained `hey` program to generate HTTP traffic [14]. We also changed the testbed machine to a Google Cloud Platform `c2-standard-4` virtual machine instance. This allows for all the Mininet switches and hosts to have enough resources to complete the benchmarks. The procedure of the experiments is as follows: we executed ping commands from Client 1 to Server 1 to measure the convergence times for each topology, i.e. the sum total of ping times until the round-trip latency times were consistently below 1 second. Then, we tested the round-trip times from each client to each server for 20 times, and obtained the average of the last 10 results. Then, we executed the VLC and `hey` benchmarks tests simultaneously for 5 minutes. We repeated the two benchmarks 31 times, and obtained the average of the last 30 executions. The first result was discarded to ensure that the results are from converged networks where the OpenFlow switches had their flows properly installed.

V. RESULTS AND ANALYSIS

Convergence times as defined in [3] increased as the number of nodes in the network increased, as shown in Table VII. This result is expected as the packets sent by the `ping` command takes longer to travel in larger networks, and is consistent with the results in [3]. This is why topologies from clusters with larger networks, i.e. Clusters 1 and 3 have longer convergence times than the others. We also note that the convergence time of the fat-tree topology and the mesh topology are closest to the results of Cluster 2 topologies. This is expected since the two aforementioned topologies are most similar in network size to Cluster 2 topologies.

Average round-trip times across all clients to all servers generally increased as the number of nodes in the network increased. No topology exhibited an average round trip time of more than 0.1ms. As with convergence times, Cluster 2, which contain the smallest topologies, performed slightly better than topologies in the other clusters. However, in this metric, Cluster 0 showed similar performance to Cluster 1.

TABLE VII. AVERAGE HTTP BENCHMARK RESULTS ACROSS ALL CLIENTS PER TOPOLOGY

Cluster	Topology	Convergence Time (ms)
0	Savvis	16.1
	Atmnet	19.2
	Ibm	21.1
1	Agis	55.5
	WideJpn	37.8
2	Gridnet	11.9
	Nsfnet	21.1
	Singaren	19.5
3	Janetbackbone	59.6
	Canerie	60.5
N/A	Fat Tree	21.8
	Mesh	11.5

TABLE VIII. AVERAGE ROUND-TRIP TIMES ACROSS ALL CLIENTS PER TOPOLOGY, IN MILLISECONDS

Cluster	Topology	Server 1	Server 2	Server 3	Server 4
0	Savvis	0.086	0.087	0.088	0.088
	Atmnet	0.085	0.085	0.085	0.086
	Ibm	0.078	0.078	0.078	0.078
1	Agis	0.083	0.083	0.082	0.082
	WideJpn	0.083	0.084	0.083	0.083
2	Gridnet	0.064	0.064	0.064	0.064
	Nsfnet	0.072	0.073	0.073	0.073
	Singaren	0.068	0.066	0.067	0.067
3	Janetbackbone	0.085	0.084	0.084	0.085
	Canerie	0.090	0.090	0.091	0.091
N/A	Fat Tree	0.065	0.065	0.066	0.065
	Mesh	0.066	0.067	0.066	0.066

The results of the HTTP traffic benchmarks are shown in Table IX. Generally, networks with smaller sizes had higher average total transfer bytes and transfer rate. In this experiment, the order of topology sizes, from smallest to biggest, is Cluster 2, Cluster 0, Cluster 1, and Cluster 3. The performance of HTTP traffic followed this order, with Cluster 2 topologies performing the best, followed by Cluster 0, etc. KPI analysis of each client showed that clients closest to the server showed the best performance compared to other nodes in the same network that requested for the same load type. For example, in the Canerie topology, client 17 was connected to switch 17, which is the closest node to the root node (switch 15). Client 17 was made to request for the 100kB JPEG file in server 1 (refer to Table III, IV), and outperformed all other clients that requested for the same file. This is expected since the server was connected to the furthest node in the network.

We also note that the fat-tree topology, which had 13 nodes, was comparable in HTTP performance to the topologies in Cluster 2, which had topology with 9 to 13 nodes. Its result was closest to the result of the Gridnet topology, which had 9 nodes and 20 edges. Meanwhile, the mesh topology, which is the smallest of the tested networks, has by far the fastest HTTP benchmark results out of the 12 topologies. This is more evidence that network size might be the key difference in terms of comparing HTTP performance under the CBQ QoS system in different networks. This result is within expectations since traffic within larger networks require more hops on average to reach various destinations.

For VLC traffic, all networks showed almost identical average demux bitrates (Table X). This is expected behavior, and a good sign that the QoS provisioning is working, as the main goal of this QoS framework is to provide stable connection and streaming of higher-priority traffic such as video, while the network is under load from other best-effort traffic. Per-client analysis of the raw data shows that for each network, all clients who accessed the same videos had similar demux bitrate, which further proves that the goal of all clients being able to have good access to video traffic has been achieved.

TABLE IX. AVERAGE HTTP BENCHMARK RESULTS ACROSS ALL CLIENTS PER TOPOLOGY

Cluster	Topology	Total Transferred (bytes)	Transfer rate (KB/s)
0	Savvis	960,048,736	3145.49
	Atmnet	1,365,445,427	4416.21
	Ibm	1,486,840,762	4801.51
1	Agis	1,199,376,942	3840.22
	WideJpn	973,259,047	3146.77
2	Gridnet	2,071,584,230	6628.80
	Nsfnet	1,582,198,537	5058.81
	Singaren	3,366,646,614	10521.1
3	Janetbackbone	733,299,798	2382.74
	Canerie	950,662,835	3113.37
N/A	Fat Tree	2,449,880,430	7530.39
	Mesh	6,033,885,718	18298.7

TABLE X. AVERAGE VLC BENCHMARK RESULTS ACROSS ALL CLIENTS PER TOPOLOGY

Cluster	Topology	Demux Bytes	Average demux bitrate (Kbits/s)
0	Savvis	46,536,544	1223.697
	Atmnet	46,552,912	1224.813
	Ibm	46,602,219	1231.056
1	Agis	46,545,751	1223.563
	WideJpn	46,534,391	1223.238
2	Gridnet	42,372,121	1225.904
	Nsfnet	44,085,597	1225.904
	Singaren	41,672,110	1240.331
3	Janetbackbone	41,903,446	1222.469
	Canerie	41,853,649	1222.751
N/A	Fat Tree	43,193,074	1237.598
	Mesh	41,503,534	1232.055

VI. CONCLUSION

In terms of average performance, the different topologies tested exhibited expected behavior for HTTP tests. In general, topologies of higher size had slower average transfer rate across all clients. Topologies of similar sizes in similar clusters showed similar results to each other. However, results on VLC video traffic was relatively consistent for all networks, which shows that the QoS was able to prioritize higher-priority traffic, which in our case, was video streaming traffic. Further work will include different queuing algorithms aside from the simple Class-based Queuing algorithm tested in this paper, such as Source-based Queuing and Source/destination-based Queuing. This will allow us to compare the behavior of different queuing algorithms in various network topologies.

REFERENCES

[1] 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.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] 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.

[3] H. S. Yoo and W. E. S. Yu, “Building a QoS Testing Framework for Simulating Real-World Network Topologies in a Software-defined Networking Environment.” in press, *8th International Conference on Engineering and Emerging Technologies*, Oct. 2022.

[4] 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.

[5] W. Zhao, D. Olshefski, and H. G. Schulzrinne, “Internet Quality of Service: An Overview,” 2000.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *RFC2475: An Architecture for Differentiated Service*. USA: RFC Editor, 1998.

[7] 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.

[8] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, “Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, 2018.

[9] 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.

[10] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Gonçalves,R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, “Implementing Quality of Service for the Software Defined Networking Enabled Future Internet,” *in 2014 Third European Workshop on Software Defined Networks*, Sep. 2014, pp. 49–54, ISSN: 2379-0369.

[11] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelem, “Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking,” in *2013 Second European Workshop on Software Defined Networks*. Berlin, Germany: IEEE, Oct. 2013, pp. 81–86.

[12] D. W. Sudiharto, F. A. Yulianto, and A. N. Arista, “Comparative analysis of voice over internet protocol (VoIP) quality on priority queue (PQ) and class-based queue (CBQ) management system using link-sharing mechanism setting,” in *2015 3rd International Conference on Information and Communication Technology (ICoICT).* Nusa Dua, Bali, Indonesia: IEEE, May 2015, pp. 419–424.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[14] J. Dogan, “rakyll/hey,” Sep. 2022, original-date: 2016-09-02T10:24:09Z.