ENHANCING AND MEASURING THE PERFORMANCE IN SOFTWARE DEFINED NETWORKING

[1]Md. Alam Hossain, [1]Mohammad Nowsin Amin Sheikh,[2,*]Shawon S. M. Rahman, [1]Sujan Biswas, and [1]Md. Ariful Islam Arman

[1]Dept. of Computer Science & Engineering, Jessore University of Science and Technology, Jessore, Bangladesh
[2]Associate Professor, Dept. of Computer Science & Engineering, University of Hawaii-Hilo, 200 W. Kawili Street, Hilo, HI 96720, USA

ABSTRACT

Software Defined Networking (SDN) is a challenging chapter in today's networking era. It is a network design approach that engages the framework to be controlled or 'altered' adroitly and halfway using programming applications. SDN is a serious advancement that assures to provide a better strategy than displaying the Quality of Service (QoS) approach in the present correspondence frameworks. SDN etymologically changes the lead and convenience of system instruments using the single high state program. It separates the system control and sending functions, empowering the network control to end up specifically. It provides more functionality and more flexibility than the traditional networks. A network administrator can easily shape the traffic without touching any individual switches and services which are needed in a network. The main technology for implementing SDN is a separation of data plane and control plane, network virtualization through programmability. The total amount of time in which user can respond is called response time. Throughput is known as how fast a network can send data. In this paper, we have design a network through which we have measured the Response Time and Throughput comparing with the Real-time Online Interactive Applications (ROIA), Multiple Packet Scheduler, and NOX.

KEYWORDS

Software Defined Networking, SDN, Quality of Service, QoS, Real-time Online Interactive Application, ROIA, Network Operating System, NOX, CES, MPLSTE, Switch Capacity, Number of Queues Impact, QoE Evaluation, Bandwidth Isolation

1. INTRODUCTION

SDN allows network operators to manage networking components using software on an external server [1]. The SDN transport network provides abstraction in three fields. It is done by the forwarding element (FE) and the control element (CE) between the networking architectures. Among the many central regulators, the distribution of control software from multiple packet forwarding nodes has been proposed to improve the flexibility of new services (i.e. virtual private network, overlays networking, content distribution, and cloud computing); standardized programmable APIs, and credibility among integrated IP networks [1,2,3,4,5].The installation of control software in a few controller nodes remotely from the forwarding elements reduces the software complexity of numerous forwarding elements and increases the overall reliability of the network [6]. SDN makes the introduction of a new vendor operating system much easier. It allows users to create plug-ins to connect control bridges to improve hardware, without changing the control hardware. Real-time Online Interactive Applications (ROIA), e.g., multiplayer online games and simulation-based e-learning, internet applications are top Internet applications that

DOI: 10.5121/ijcnc.2018.10502

claim the highest Quality of Service (QoS) on the underlying networks [21,31]. This demand depends on the number of users and the actual application state and, therefore, is changed at runtime. Some SDN-based jobs are targeted to meet the needs of network resources, policy-based network provisioning is targeted [7, 8,22], whereas wide area networks (WANs) are targeted to traffic engineering [9, 10, 20]. Dynamic allocations of network resources are also required in data centers and many studies deal with these challenges. For example, an Open Flow-based algorithm [17] for allocation of bandwidth resources in Virtual Machine is presented in data centers [11] when [12] the author describes a platform for coordinating the provision of calculation, storage and network resources in the data centers. The Network Operating System (NOX) does not work on the network itself; it provides a programming interface with high-level objects (such as CPU processing power, disk storage volume, memory, link power, etc.) of network resources, enabling network application programs to run securely and efficiently over a wide variety of network programs[28].

2. EXISTING SYSTEM AND IT'S PROBLEM

2.1 ROIA

Real-time Online Interactive Applications (ROIA) are a possible network application connected with a number of users which could interact with applications and the truth, for example, a replication to a user's action transpires virtually immediately. Due to a large, variable user with intensive and dynamic interaction, ROIA claims high Quality of Services (QoS) of low networks. In addition, these needs can change constantly; the number of users and the actual application depends on the state: In a shooter game, a high packet loss in a warring kingdom can be fatal consequences on QoS [21]. It is less relevant when a player is exploring the landscape.

The ROIA applications are divided into two parts, a static and a dynamic part. The static part has a non-variable and landscape objects. Playing non-game controlled by one of the other dynamic parts in the server. These objects can change their status at any time. Figure 1 shows the structure of an ROIA. This architecture serves only one ROIA processed ROIA client. But a group of ROIA processes is distributed among different machines. In an approximate loop processing, ROIA is reconsidered in a real state, is known as the real-time loop [24]. There are three main steps for a single loop repeat. First, the user sends the input through the network and sends it and gets cordially via the ROIA process. Then, to calculate the application state, we can apply user input and logic to the current state. After that, the loop is transferred to the client while updating.

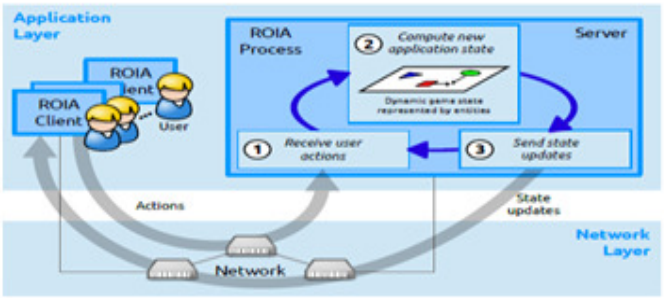

Figure 1.Structure of an ROIA and its real-time loop

Figure 2 shows the graph of the calculation of Response Time with ROIA [16] of Table 1.

Table 1. Calculation of Response Time and Throughput with ROIA

Number of Operations	Response Time (ROIA) ms	Throughput (ROIA)
5	1.03	0.97087
10	1.19	0.84033
15	1.22	0.81967
20	1.35	0.74074
25	1.29	0.77519
30	1.07	0.93457
35	1.48	0.67567
40	1.21	0.82644
45	1.34	0.74626
50	1.09	0.91743
55	1.42	0.70422
60	1.3	0.76923
65	1.15	0.86959
70	1.45	0.68965

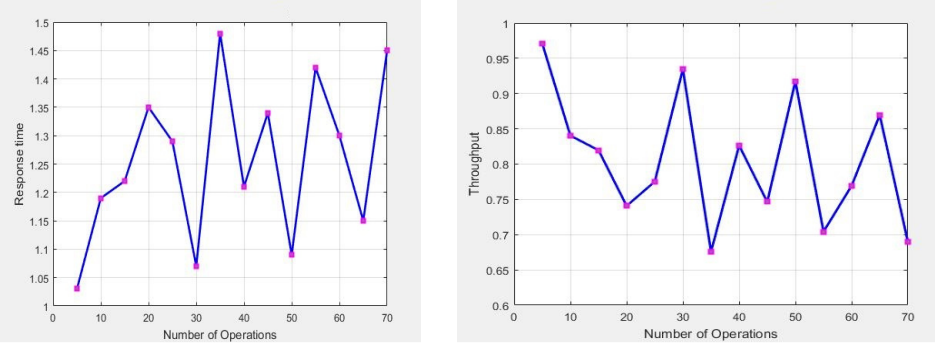

Figure 2. Response Time and Throughput of ROIA

2.2 MULTIPLE PACKET SCHEDULER

The Open Flow data path plus QoS modules is for the QoS Flow data path. This datapath is a utility space implementation where queues are located in the kernel space. The QoS module opens a channel with the kernel through the Net link and Packet socket families to connect both utilize and kernel space. Thus, the packet schedulers can be instantiated to enable traffic shaping and enqueueing of flows. The components called Traffic Shaping, Packet Schedulers and enqueueing that constructs the QoS module of the QoS Flow data path, and their relationships are illustrated in Figure 3.

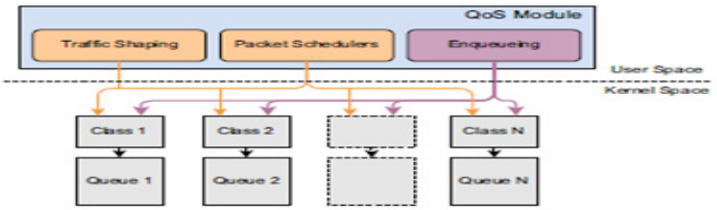

Figure 3. QoS module which has been added to the standard Open Flow data path

Traffic Shaping and Packet Schedulers: These components use Net link socket family to manipulate OFPT_QOS_QUEUEING_DISCIPLINE message type, which is a new extension of the message to represent the QoS message in OF protocol.

Hence, the Traffic Shaping and Packet Schedulers components administer the QoS messages receipt from control plane by splitting the bandwidth size in queues and by attaching or detaching packet schedulers for these queues, respectively. To establish a connection with the kernel, these components open a Net link socket channel and send a Net link message through it. The Net link message is the type of message that Linux kernel accepts for network resources management. In this way, the QoS messages are mapped to Net link messages.

Enqueueing. It is the component responsible to operate OFPT_FLOW_MOD messages of the OF protocol. This message modifies the state of the flow table, where each entry contains header fields, counters, and actions for matching packets or flow packets. The enqueueing mechanism maps, flow to queues using the skip-> priority of kernel data structure called sk_buff. This configuration is done through the use of the SO_PRIORITY option of the Packet socket family [33]. Sinceuserspace cannot access such data structure directly. The QoS development strategy for OF enabling networks to overcome packet scheduling issues. The main goal of QoS Flow is to allow control of multiple packet schedulers. In another word, QoS Flow brings the traffic control of Linux to become part of ONF networks.[30] Our proposal extends the OF protocol 1.0 and the standard datapath based on it. This way, developers can deploy their own application to enable, for instance, a control of bandwidth-on-demand with one or more packet schedulers on the network. Currently, QoS Flow provides control of the following packet schedulers: HTB (Hierarchical Token Bucket) [25], RED (Randomly Early Detection) [26], and SFQ (Stochastic Fairness Queuing) [27].Currently, QoS Flow controls the following packet schedulers: HTB, SFQ, and RED where the HTB is a classfull, while SFQ and RED are classless queuing discipline. Thus, the current QoS Flow features come from these Linux kernel packet schedulers.

HTB: Allows splitting bandwidth size of the network. By default, the Linux kernel automatically attaches a FIFO packet scheduler to each bandwidth segment. It creates logical links which are slower than a physical link.

SFQ: Belongs to fair queuing algorithms. The SFQ schedules the packet transmission based on information about the IPv4/v6 source and destination address, and TCP/UDP source port to assign each flow to each hash bucket, on the enqueueing phase.

RED: It drops packets in a queue gradually. It performs a tail drop like FIFO, but smartly. Such a packet scheduler has a threshold value to mark packets to be discarded after queue length becomes greater than the threshold value.

Figure 4 shows the graph of the calculation of Response Time and Throughput with Multiple Packet Scheduler [18] of Table 2

.

Table 2. Calculation of Response Time and Throughput with Multiple Packet Schedulers

Number of Operations	Response Time (HTB) ms	Response Time (SFQ) ms	Response Time (RED) ms	Throughput (HTB) ms	Throughput (SFQ) ms	Throughput (RE
5	1.28	0.1	0.55	.12206	.15625	0.028
10	2.56	0.2	1.1	.244125	.3125	0.056
15	3.84	0.3	1.65	.36618	.46875	0.085
20	5.12	0.4	2.2	.48825	.625	0.113
25	6.4	0.5	2.75	.6103125	.78125	0.142
30	7.68	0.6	3.3	.732375	.9375	0.170
35	8.96	0.7	3.85	.8544375	1.09375	0.198
40	10.24	0.8	4.4	.9765	1.25	0.227
45	11.52	0.9	4.95	1.09856	1.40625	0.255
50	12.8	1	5.5	1.220625	1.5625	0.284
55	14.08	1.1	6.05	1.34268	1.71875	0.312
60	15.36	1.2	6.6	1.46472	1.875	0.340
65	16.64	1.3	7.15	1.58678	2.03125	0.369
70	17.92	1.4	7.7	1.70884.	2.1875	0.397

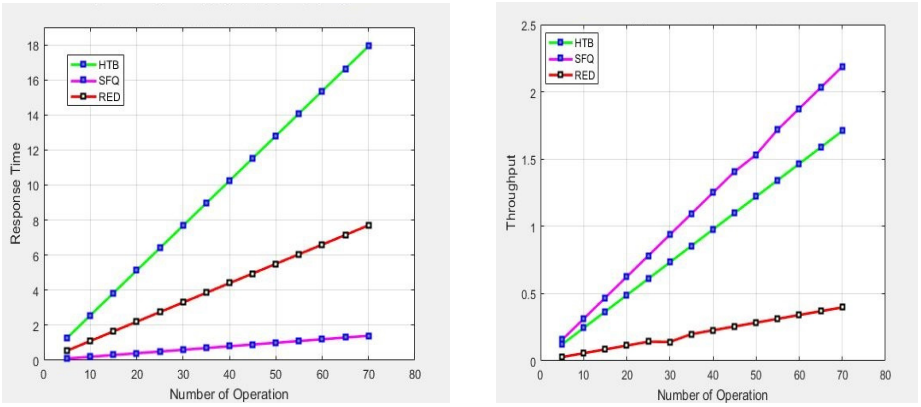

Figure4. Response Time and Throughput of Multiple Packet Schedulers

2.3 NOX

Network Operating System (NOX) management applications are built as a central program in order to engender high caliber relinquishments of network resources in contrast to the algorithms distributed on low-level addresses [23, 24]. The network operating system does not manage the network itself. It provides a programming interface with high calibers of network resources (e.g. recollection, disk storage volume, CPU processing puissance, disk storage volume, link potency, etc.) that enables network application programs to perform involutes tasks safely and efficiently in a wide range of networking technologies [23]. The NOX, however, fails in giving the indispensable functions for QoS-assured Software Defined Networking (SDN) [22, 25, 35] accommodation provisioning on the bearer grade provider internet, such as QoS-vigilant virtual network seating, end-to-end network QoS quantification, and cooperation among control elements in another domain network. Figure 5 shows the graph of the calculation of Response Time and Throughput with NOX [19] of Table 3.

Table 3. Calculation of Response Time and Throughput with NOX

Number of Operations	Response Time (NOX) ms	Throughput (NOX) ms
5	0.7948	0.09
10	0.983	0.091
15	0.8542	0.08
20	0.808	0.075
25	0.888	0.0859
30	0.899	0.0848
35	0.9585	0.079
40	0.97	0.082
45	0.787	0.072
50	0.9095	0.0797
55	0.755	0.0976
60	0.7777	0.0776
65	0.842	0.0923
70	0.7888	0.0948

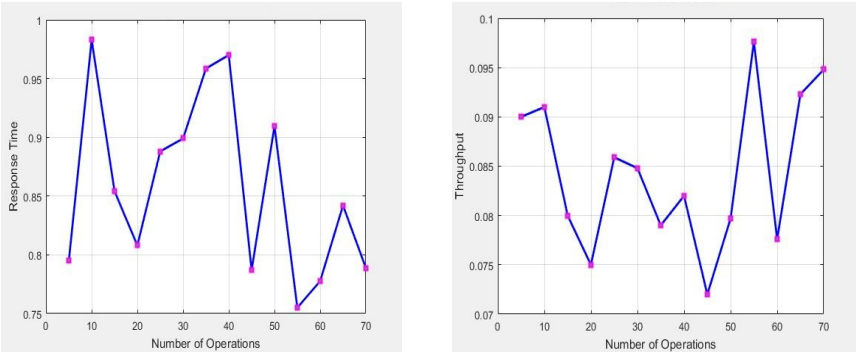

Figure 5. Response Time and Throughput with NOX

3. RELATED WORK

SDN allows network operators to manage networking components using software on an external server [4, 41]. The SDN Transport Network (Distribution Status, Forwarding, and Configuration) provides an abstraction in three fields, the simplest way to create simplicity. It is done by forwarding element (FE) and the control element (CE) between the networking architecture. Among the many central regulators, the distribution of control software from multiple packet forwarding nodes has been proposed to improve the flexibility of new services (i.e. virtual private network, overlays networking, content distribution, and cloud computing); standardized programmable APIs, and credibility among integrated IP networks[4-7].Since installing distribute, forward components, central control software on several remote control nodes reduces the software complexity of many forwarding components, and it increases network overall fidelity [5]. SDN makes the introduction of a new vendor operating system much easier. It allows users to create plug-ins to connect control bridges to improve hardware, without changing the control hardware, without changing the hardware included.

Real-time Online Interactive Applications (ROIA), e.g., multiplayer online games and simulation-based e-learning, Internet applications are top Internet applications that claim high-quality services (QoS) underlying networks. This demand depends on the number of users and the actual application state and, therefore, is changed at runtime.[32] Common networks have very limited potential to influence network behavior to meet dynamic QoS requirements, as most ROIA uses underlying networks based on a best-effort basis.

Some SDN-based jobs are targeted to meet the needs of network resources, policy-based network provisioning is targeted at [11, 12, 37], whereas wide area networks (WANs) are targeted to traffic engineering [13,14, 20]. Dynamic allocation of network resources is also required in data centers and many studies deal with this challenge. For example, an OpenFlow-based algorithm [17] for allocation of bandwidth resources in Virtual Machine is presented in Data Centers [15], when [16] the author describes a platform for coordinating the provision of calculation, storage and network resources in the data centers. However, the most relevant work focuses on the service logic for QoS-aware resource provisioning, finding out the details of how managed and privileged the network resources are in resources.

The Network Operating System (NOX) does not work on the network itself; it provides the NOX Carrier Grade Production Internet, such as QoS-aware virtual network embedding, end-to-end network QoS evaluation, QoS-guaranteed software scheduled networking (SDN) [4] fails to provide the functions required to provide the service, and co-operation between control elements of other domain networks.

4. OVERVIEW OF THE PROPOSED SYSTEM

We have designed a QoS module for getting better performance during packet passing. We can test our designed QoS module on the basis of response time, throughput, and bandwidth isolation and switch capacity.

In our proposed architecture there are three networks. Each network consists of three routers and three users or hosts connected with each other. In every network, the routers are interconnected with each other. Three networks are connected with the SDN controller, which is QoS performance monitor also. After interconnecting the routers of each network, the network will be connected to each other. By interconnecting these networks, all hosts and routers will be interconnected with themselves.

We implemented our proposed architecture in mininet environment in Linux operating system. To implement this architecture, at first, we have to set up mininet environment of our machine. We have created a topology to build up the architecture with the help of the Python language. We define the hosts and routers, add the hosts by self-addhost function and add the routers by self-add switch function. After combining all the routers and hosts we have checked all the hosts.[29] For checking the establishment of the architecture we use the ping command in mininet environment.

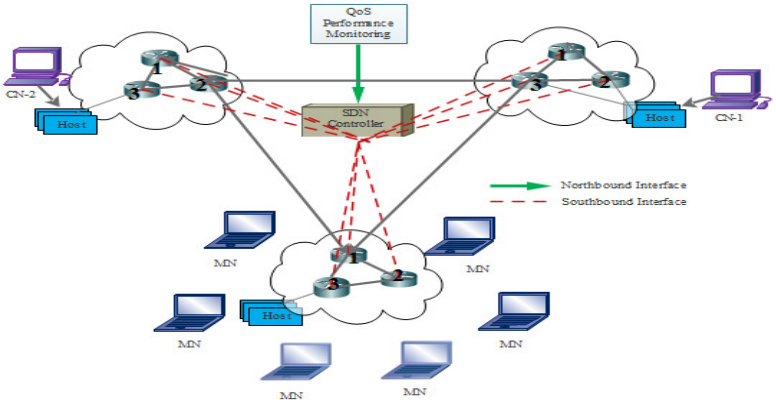

Figure 6: Developed_QoS_module

Table 4: Calculation of Response Time and Throughput with Developed QoS module

Number of Operations	**Response Time (Developed_QoS_module) ms**	**Throughput (Developed_QoS_module) ms**
5	0.7104	0.0514
10	0.783	0.0792
15	0.8062	0.0612
20	0.7372	0.0488
25	0.797	0.0606
30	0.809	0.0646
35	0.7736	0.062
40	0.892	0.059
45	0.6482	0.06
50	0.8104	0.0582
55	0.666	0.0466
60	0.6454	0.0556
65	0.703	0.0752
70	0.7054	0.0738

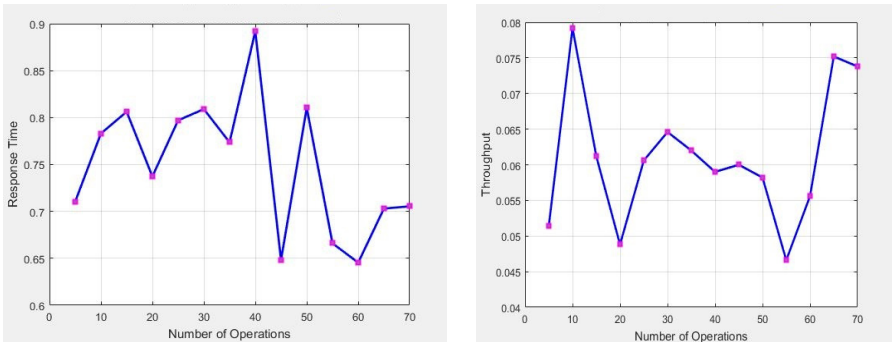

Figure 7. Response Time of Proposed System

Pseudo code

1. Take the number of hosts and routers as n and r
2. Connect n hosts and r routers
3. for i=1 to n
4. for j=1 to r
5. Self. Add link(host[i],router[j])
6. End

5. PERFORMANCE ANALYSIS

5.1 COMPARISON OF THROUGHPUT

We can calculate the performance in two cases. The first case is our developed module is better and another case is existing systems are better than our system. We can calculate how much better our developed module than other systems by the following equation:

$$\text{Performance (p)} = m/n$$

Where m=sum of throughput of Developed QoS module and n=sum of throughput of existing systems.

We also calculate how much better other systems than our developed module by the following equation:

$$\text{Performance (p)} = n/m$$

Where n=sum of throughput of Developed QoS module and m=sum of throughput of existing systems.

We executed 70 operations in our Developed QoS module. In comparison with a throughput of ROIA and our Developed QoS modules, the throughput of the Developed QoS module is 13.17x better than the throughput of ROIA. The throughput of the ROIA and Developed QoS module is given below.

The throughput of Developed_QoS_module is 14.97x and 19.16x higher in comparison with HTB and SFQ packet schedulers. In case of a RED packet scheduler, the throughput of RED is 1.53x higher for the first 10 packets in comparison with our Developed QoS module. After passing of the first 10 packets our Developed QoS modules' throughput is 3.99x better than a RED packet scheduler. The throughput of Developed_QoS_module is 1.38x better than the throughput of NOX.

Table 5: Comparison of Throughput

Number of Operations	Throughput (ROIA) ms	Throughput (HTB) ms	Throughput (SFQ) ms	Throughput (RED) ms	Throughput (NOX) ms	Throughput (Proposed System) ms
5	0.97087	0.12206	0.15625	0.0284	0.09	0.0514
10	0.84033	0.244125	0.3125	0.05681	0.091	0.0792
15	0.81967	0.36618	0.46875	0.08522	0.08	0.0612
20	0.74074	0.48825	0.625	0.11363	0.075	0.0488
25	0.77519	0.610312	0.78125	0.14204	0.0859	0.0606
30	0.93457	0.732375	0.9375	0.17045	0.0848	0.0646
35	0.67567	0.854437	1.09375	0.19886	0.079	0.062
40	0.82644	0.9765	1.25	0.22727	0.082	0.059
45	0.74626	1.09856	1.40625	0.25567	0.072	0.06
50	0.91743	1.22062	1.5625	0.28408	0.0797	0.0582
55	0.70422	1.34268	1.71875	0.31249	0.0976	0.0466
60	0.76923	1.46472	1.875	0.34090	0.0776	0.0556
65	0.86959	1.58678	2.03125	0.36931	0.0923	0.0752
70	0.68965	1.70884	2.1875	0.39772	0.0948	0.0738

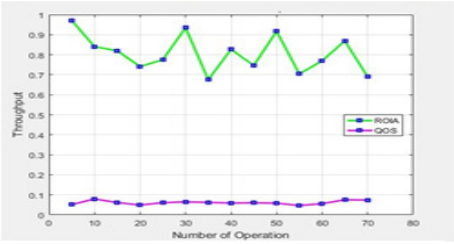

Figure 8. Comparison of Throughput between developed QoS module and ROIA

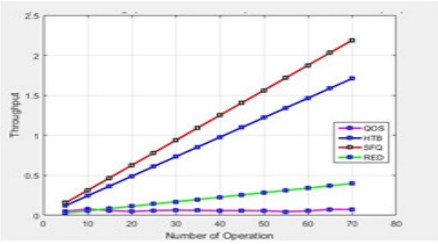

Figure 9. Comparison of Throughput between developed QoS module and multiple packet schedulers.

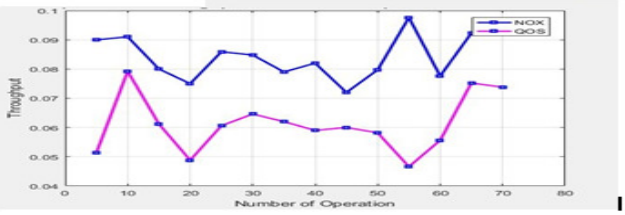

Figure 10. Comparison of Throughput between developed QoS module and NOX

5.2 Comparison of Response Time

In a comparison of response time between ROIA and our Developed QoS module, the response time of the Developed QoS module is 1.677x higher than the response time of ROIA. In a comparison of response time with multiple packet schedulers, there are three packet schedulers we have compared with our Developed QoS module. They are HTB, SFQ and RED packet scheduler. The response time of the Developed QoS module is 1.7x better than an HTB packet scheduler. In case of a RED packet scheduler, the response time of first 6 packets is 1.2x better than Developed QoS module and for other packet passing response time of Developed QoS,themodule is 1.37x higher than a RED packet scheduler. We have compared the response time with SFQ packet scheduler, the response time of first 41 packets passing is 7.70x higher than

a Developed QoS module. The response time of next 29 packets in Developed QoS module is 8.74x better than SFQ packet scheduler. The response time of Developed QoS_module is 1.07x higher than the response time of NOX.

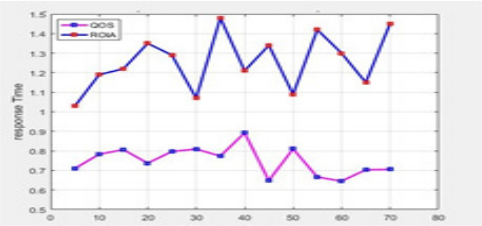

Figure 11. Comparison of Response time between developed_QoS_module and ROIA

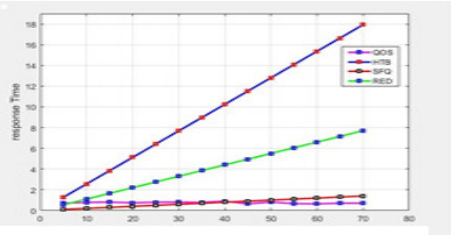

Figure 12. Comparison of Response time between developed_QoS_module and multiple packet schedulers.

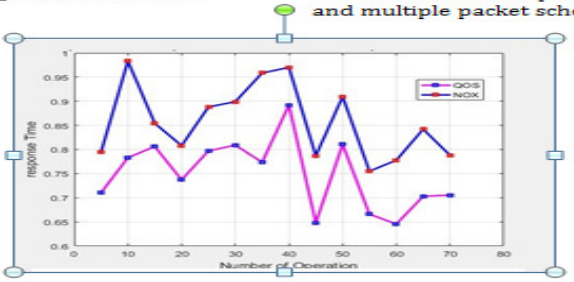

Figure 13. Comparison of Response time between developed_QoS_module and NOX

Table 6. Comparison of Response Time

Number of Operations	Response Time (ROIA) ms	Response Time (HTB) ms	Response Time (SFQ) ms	Response Time (RED) ms	Response Time (NOX) ms	Response Time (Developed_QoS_module) ms
5	1.03	1.28	0.1	0.55	0.7948	0.7104
10	1.19	2.56	0.2	1.1	0.983	0.783
15	1.22	3.84	0.3	1.65	0.8542	0.8062
20	1.35	5.12	0.4	2.2	0.808	0.7372
25	1.29	6.4	0.5	2.75	0.888	0.797
30	1.07	7.68	0.6	3.3	0.899	0.809
35	1.48	8.96	0.7	3.85	0.9585	0.7736
40	1.21	10.24	0.8	4.4	0.97	0.892
45	1.34	11.52	0.9	4.95	0.787	0.6482
50	1.09	12.8	1	5.5	0.9095	0.8104
55	1.42	14.08	1.1	6.05	0.755	0.666
60	1.3	15.36	1.2	6.6	0.7777	0.6454
65	1.15	16.64	1.3	7.15	0.842	0.703
70	1.45	17.92	1.4	7.7	0.7888	0.7054

6. CONCLUSION

Software Defined Networking is an emerging topic for the modern era. It is an idea which has recently reignited the interest of network researchers for programmable networks. Enabling added-value services are the main target for this work. Not only this but also ensuring the security [34][35][36]is another purpose of this work. The SDN enables an easy and flexible realization of existing dynamic Quality of Service (QoS) mechanisms in today's communication networks. Although SDN and, in particular, Open Flow claims to provide a standardized interface, the

existing diversity of Open Flow enables switches leads to varying behavior for the same QoS mechanisms. We will improve Quality of Services (QoS) in SDN by building an architecture which will be implemented in any network emulator.

7. FUTURE WORK

In this paper, we have used with two parameters, such as Response Time and Throughput.In the future implementation; we aim to use Switch Capacity, Number of Queues Impact, QoE Evaluation, and Bandwidth Isolation.

REFERENCES

[1] "Improving QoS in Real-Time Internet Applications: From Best-Effort to Software-Defined Networks - IEEE Xplore Document."10 April 2014.

[2] "Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking - IEEE Xplore Document." 12 December 2013.

[3] Mudit Saxena, and Dr. Rakesh Kumar."A Recent Trends in Software Defined Networking (SDN) Security." International Conference on Computing for Sustainable Global Development (INDIACom).on 2016 IEEE.

[4] Natasha Gude et al., "NOX: Towards an Operating System for Networks," editorial note submitted to CCR.

[5] Arsalan Tavakoli et al, "Applying NOX to the Datacenter," in Proc. Of SIGCOMM Hotnet 2009.

[6] Dimitri Staessens et al., "Software Defined Networking: Meeting Carrier Grade Requirements," in Proc. of IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), 2011.

[7] P. Georgopoulos, Y. Elkhatib, M. Broadbent et al., "Towards network wide QoE fairness using OpenFlow-assisted adaptive video streaming," in Proc. of the 2013 ACM SIGCOMM Workshop on Future Human- Centric Multimedia Networking (FhMN 2013), Hong Kong, China, 2013, pp. 15–20.

[8] T. Zinner, M. Jarschel, A. Blenk et al., "Dynamic application-aware resource management using software-defined networking: implementation prospects and challenges," in Proc. of the 2014 IEEE Network Operations and Management Symposium (NOMS '14), Krakow, Poland, 2014, pp. 1–6.

[9] Lazaris, D. Tahara, X. Huang et al., "Tango: simplifying SDN control with automatic switch property inference, abstraction, and optimization," in Proc. of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT), Sydney, Australia, 2014, pp. 199–212.

[10] M. Kuzniar, P. Peresini, and D. Kostic, "What you need to know about SDN control and data planes," EPFL, Lausanne, Switzerland, Tech. Rep. EPFL-REPORT-199497, 2014.

[11] V. Mann, A. Vishnoi, A. Iyer et al., "VMPatrol: dynamic and automated QoS for virtual machine migrations," in Proc. of the 8th International Conference on Network and Service Management (CNSM), Las Vegas, USA, 2012, pp. 174–178.

[12] Z. Bozakov and A. Rizk, "Taming SDN controllers in heterogeneous hardware environments," in Proc. of Second European Workshop on Software Defined Networks (EWSDN), Berlin, Germany, 2013, pp. 50 – 55.

[13] M. Kuzniar, P. Peresini, and D. Kostic, "What you need to know about sdn flow tables," in Passive and Active Measurement, ser. Lecture Notes in Computer Science, J. Mirkovic and Y. Liu, Eds. Springer International Publishing, 2015, vol. 8995, pp. 347–359.

[14] P. M. Mohan, D. M. Divakaran, and M. Gurusamy, "Performance study of TCP flows with QoS-supported OpenFlow in data center networks," in Proc. of the 19th IEEE International Conference on Networks (ICON), Singapore, Singapore, 2013, pp. 1–6

[15] Nguyen-Ngoc, S. Lange, S. Gebert et al., "Investigating isolation between virtual networks in case of congestion for a Pronto 3290 switch," in Proc. of the Workshop on Software-Defined Networking and Network Function Virtualization for Flexible Network Management (SDNFlex 2015), Cottbus, Germany, 2015.

[16] Bari, M.F., Chowdhury, S.R., Ahmed R., Boutaba, R.: PolicyCop: an autonomic QoS policy enforcement framework for software defined networks. In: IEEE SDN for Future Networks and Services, Trento, Italy, pp. 1–7, November 2013.

[17] Egilmez, H.E., Dane, S.T., Bagci, K.T., Tekalp, A. M.: OpenQoS: an openflow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. In: Proceedings of the Signal and Information Processing Association Annual Summit and Conference, Hollywood, California, US, pp. 1–8, December 2012.
[18] Guo, J., Fangming, L., Haowen, T., Yingnan, L., Hai, J., John, L.: Falloc: fair network bandwidth allocation in IaaS datacenters via a bargaining game approach. In: Proceedings of the ICNP, Gotingen, Germany, pp. 1–10, October 2013.
[19] Benson, T., Akella, A., Shaikh, A., Sahu, S.: CloudNaaS: a cloud networking platform for enterprise applications. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, Cascais, Portugal (2011).
[20] Jain, S., et al.: B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Comput. Commun. Rev. 43(4), 3–14 (2013).
[21] Kim, W., et al.: Automated and scalable QoS control for network convergence. In: Proceedings of the INM/WREN, San Jose, California, US (2010).
[22] M. Betts, S. Fratini, N. Davis, R. Dolin and others, "SDN Architecture". Open Networking Foundation ONF SDN ARCH, Issue 1, June, 2014.
[23] M. Joselli et al., "An Architeture with Automatic Load Balancing for Real-Time Simulation and Visualization Systems," Journal of Computational Interdisciplinary Sciences, vol. 1, no. 3, pp. 207–224, 2010.
[24] Bert Hubert, Thomas Graf, Gregory Maxwell, Remco Van Mook, Martijn Van Oosterhout, Paul B. Schroeder, Jasper Spaans, and Pedro Larroy. Linux Advanced Routing & Traffic Control HOWTO. Linux Advanced Routing & Traffic Control, http://lartc.org/, April 2004.
[25] Paul E McKenney. Stochastic fairness queueing. In INFOCOM'90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies.'The Multiple Facets of Integration'. Proceedings. IEEE, pages733–740. IEEE, 1990.
[26] "On Scalability of Software-Defined Networking - IEEE Xplore Document." 14 February 2013.
[27] Alexander So." Survey on Recent Software-Defined Network Cross-Layer Designs." https://www.researchgate.net/publication/311953252_Survey_on_Recent_Software.Defined_Network_Cross-Layer_Designs on December 2016.
[28] Cisco 2016. Unicast flooding in switched campus networks; http:// www. Cisco.com/c/n/us/support/docs/switches/catalyst/6000-series-switches/23563-143.html.
[29] ONF.OpenFlow table type patterns, Open Network-ing Foundation, Tech. Rep.Available from: https://www.opennetworking.org/images/stories/downloads/sdnresources/onfspecifications/openflow/OpenFlow,[Accessed on: March 9, 2018].
[30] Haleplidis E, Denazis S, Pentikousis K, Denazis S,Salim JH, Meyer D, Koufopavlou O.SDN layersand architecture terminology, Internet draft, Internet engineering task force. Available from: https://www.ietf.org/id/draft-irtfsdnrg-layer-terminology-02.txt,[Accessed on: February 20, 2018].
[31] ON.LAB. ONOS: Open Network Operating System. In ONS, 2017.
[32] H. Howard, D. Malkhi, and A. Spiegelman. Flexible Paxos: Quorum intersection revisited. CoRR, abs/1608.06696, 2016.
[33] Loukaka, Alain and Rahman, Shawon; "Discovering New Cyber Protection Approaches From a Security Professional Prospective"; International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.4, July 2017
[34] Al-Mamun, Abdullah, Rahman, Shawon and et al;" Security Analysis of AES and Enhancing its Security by Modifying S-Box with an Additional Byte "; International Journal of Computer Networks & Communications (IJCNC), Vol.9, No.2, March 2017
[35] Opala, Omondi John; Rahman, Shawon; and Alelaiwi, Abdulhameed; "The Influence of Information Security on the Adoption of Cloud computing: An Exploratory Analysis", International Journal of Computer Networks & Communications (IJCNC), Vol.7, No.4, July 2015
[36] Halton, Michael and Rahman, Syed (Shawon); "The Top 10 Best Cloud-Security Practices in Next-Generation Networking"; International Journal of Communication Networks and Distributed Systems (IJCNDS), Vol. 8, Nos. ½, 2012, Pages:70-84
[37] Schuett, Maria and Rahman, Syed (Shawon); "Information Security Synthesis in Online Universities"; International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.5, Sep 2011

AUTHORS

Md. Alam Hossain is working as an Assistant Professor at the Department of Computer Science & Engineering in Jessore University of Science & Technology, Bangladesh. He completed his B.Sc and M.Sc (Thesis) in Computer Science & Engineering from Islamic University, Bangladesh. Currently, he is pursuing Ph.D. on Cloud Computing Security.

Mohammad Nowsin Amin Sheikh is working as an Assistant Professor at the Department of Computer Science & Engineering in Jessore University of Science & Technology (JUST), Jessore, Bangladesh. He completed his B.Sc. (Engg.) in Computer Science & Engineering from Jessore University of Science & Technology (JUST), Jessore, Bangladesh.

Dr. Shawon S. M. Rahman is an Associate professor in the Department of Computer Science and Engineering at the University of Hawaii-Hilo. His research interests include software engineering education, information assurance and security, web accessibility, cloud computing, and software testing andquality assurance. He has published over 100 peer-reviewed papers. He is an active member of manyprofessional organizations including IEEE, ACM, ASEE, ASQ, and UPE.

Sujan Biswas completed his B.Sc in Computer Science & Engineering from Jessore University of Science & Technology, Bangladesh.

Md. Ariful Islam Arman completed his B.Sc in Computer Science & Engineering from Jessore University of Science & Technology, Bangladesh.