Global Journal of Computer Science and Technology: E Network, Web & Security
Volume 18 Issue 3 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Quality of Service in Software Defined Networking

By Md. Alam Hossain, Mohammad Nowsin Amin Sheikh, Monishanker Halder, Sujan Biswas & Md. Ariful Islam Arman
Jessore University of Science and Technology

Abstract- Software Defined Networking (SDN) promises to provide a powerful way to introduce Quality of Service (QoS) concepts in today's communication networks. In SDN the behavior and the functionality of the network devices is programmatically modified using a single high-level program. Software Defined Networking (SDN) instantiation OpenFlow is designed according to these properties. The realization of the Quality of Service (QoS) concepts becomes understandable with SDN in a convenient way. This paper focuses on the existing architectures parameters such as response time, switch capacity and bandwidth isolation and we evaluate these parameters here. Although concepts of QoS are well researched, it is hard to understand that due to high implementation complexity and realization costs. OpenFlow as the best-known SDN standard so far defines a standard protocol for network control. These observations of switch variety may provide SDN application developer's insights when realizing QoS concepts in an SDN-based network.

Keywords: *SDN, QoS, ROIA, NOX, ForCES, MPLSTE, RSVP, FE, CE, HTB, SFQ, RED, QoS.*

GJCST-E Classification: *C.2.m*

Strictly as per the compliance and regulations of:

Quality of Service in Software Defined Networking

Md. Alam Hossain [α], Mohammad Nowsin Amin Sheikh [σ], Monishanker Halder [ρ], Sujan Biswas [ω] & Md. Ariful Islam Arman[¥]

Abstract- Software Defined Networking (SDN) promises to provide a powerful way to introduce Quality of Service (QoS) concepts in today's communication networks. In SDN the behavior and the functionality of the network devices is programmatically modified using a single high-level program. Software Defined Networking (SDN) instantiation OpenFlow is designed according to these properties. The realization of the Quality of Service (QoS) concepts becomes understandable with SDN in a convenient way. This paper focuses on the existing architectures parameters such as response time, switch capacity and bandwidth isolation and we evaluate these parameters here. Although concepts of QoS are well researched, it is hard to understand that due to high implementation complexity and realization costs. OpenFlow as the best-known SDN standard so far defines a standard protocol for network control. These observations of switch variety may provide SDN application developer's insights when realizing QoS concepts in an SDN-based network.

Keywords: SDN, QoS, ROIA, NOX, ForCES, MPLSTE, RSVP, FE, CE, HTB, SFQ, RED, QoS.

I. Introduction

SDN provides network functionality and also the behavior of network devices. The data-plane is forward through the network, such as packets and the hardware that is used to forward it, such as, switches. The control-plane represents all logic and devices that are responsible for deciding how and to where data in the data-plane is to be sent. In SDN, network operator can manage networking elements by running software on an external server. We can easily understand SDN against traditional network by a simple example, suppose if we want to deliver a packet in the network it has to change its route multiple times for finding the optimal path. But in SDN it automatically traces the entire possible and shortest route for delivering the packets. By separating the control plane from data plane in SDN, some controller increased its flexibility in deploying new services (e.g., virtual private network, cloud computing), programmability in open API, reliability in converged IP network. In few controllers installing control software remotely from forwarding element reduces the complexity of the forwarding element but increases the dependability of the network.

Real-time Online Interactive Applications (ROIA) is connecting high numbers of user applications such as multiplayer online games, simulation base e-learning etc. In ROIA the reaction of user happens virtually and immediately. ROIA provides High Quality of Service on underlying network. Resource Reservation Protocol (RSVP) is one kind of traditional technique for controlling QoS for reserving network bandwidth. It is mainly static by nature and not suitable for changing demands of ROIA dynamically. There is a problem for SDN is to design the Northbound API. It defines the communication between the controller and high-level program [1].

Packet scheduling is essential for entire supporting applications on Software-Defined Networking (SDN) model. However, on OpenFlow/SDN, QoS is only performed with bandwidth guarantees and by a well-known FIFO scheduling. QoSFlow module adds extensions to the standard software switch (datapath) of Open Flow 1.0. During the starting time of the QoSFlow project, the specification 1.0 of Open Flow was the latest stable version and was used in the project. Even though OF 1.3 has brought a new mechanism for rate limiting, but as well as the Open Flow 1.0, we are able to use FIFO instead of other packet schedulers to achieve different treatments to the packets. The Open Flow datapath plus QoS modules form the QoSFlow datapath [2]. This datapath is a user space implementation where queues are located in the kernel space. The QoS module opens a channel with the kernel through Netlink and Packet socket families to connect both user 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 QoSFlow datapath.

Network operating systems (NOX) applications will be written as centralized programs on high level of instability in network resources, unlike algorithms distributed from lower back [3, 4] applications. The network operating system does not manage the network itself; It provides a programming interface with high level objects (such as CPU processing power, memory, disk storage volume, link

Author α σ: Assistant Professor, Dept. of Computer Science and Engineering, Jessore University of Science and Technology, Jessore.
e-mails: alamcse_iu@yahoo.com, nowsin.jstu@gmail.com
Author ρ: Lecturer, Dept. of Computer Science and Engineering, Jessore University of Science and Technology, Jessore.
e-mail: monicsejust@gmail.com
Author ω ¥: Dept. of Computer Science and Engineering, Jessore University of Science and Technology, Jessore.
e-mails: sujanbiswas1357@gmail.com, armanarif.cse@gmail.com

power, etc.) of network resources, which enables network application programs to handle secure and functional complex tasks on a wide variety of networks [3]. The NOX, however, fails in providing the obligatory functions for QoS-ensured software defined networking (SDN) [5] accommodation provisioning on carrier grade provider Internet, such as QoS-vigilant virtual network embedding, end-to-end network QoS assessment, and collaborations among control elements in other domain network.

II. Related Work

Previous work on providing QoS using Open Flow has three categories. First, studies deploying dynamic QoS in an SDN environment [6], [7], [8]. Second, studies on switch diversity [9], [10], [11], [12].Third, research on network performance resulting from QoS with OpenFlow- enabled switches [13], [14].

Some of the work done in the area of SDN based on demand provisioning of network resources, is targeted towards automated, policy-based network provisioning [15, 16], while the other is targeted towards traffic engineering across Wide Area Networks (WANs) [17, 18]. Dynamic allocation of network resources is also required inside the data centers, and many studies address this challenge. As an example, an OpenFlow-based algorithm for allocation of bandwidth resources between virtual machines in data centers is presented in [19], while in [20] the authors describe a platform for integrated provisioning of computing, storage and network resources in data centers.

However, most of the related work focuses on the service logic for QoS-aware resource provisioning, leaving out the details of how the network resources are managed and provisioned in the data plane. In some of the papers, such as [21], the authors mention that the proposed QoS architecture dependent on traffic classification and rate limiting at the edge of the network, although no description about how the reservation of logical resources inside the SDNC enforces in the forwarding devices.

The work that is closest to the one presented here is [22], which defines a data plane QoS architecture for SDN based on similar principles (i.e. a combination of queues and rate limiters), but with different constraints for the resources. The current paper contains a definition of how the resources are managed inside the SDN in order to provide deterministic QoS.

When monitoring the QoS it is an advantage that a network problem resulting in decayed performance will affect a whole class of flows that share some properties (i.e. routed through an overloaded device and use the same QoS class). It is sufficient to monitor a representative subset of the network flows which makes QoS Monitoring eligible for sampling.

III. Description of the Existing Architectures

Existing Architectures

1. ROIA
2. Multiple Packet Scheduler
3. NOX

1. ROIA

Real-time Online Interactive Applications (ROIA) connects wide number of users who interact with the applications and with each other with proper authentication, i.e., a replication to a user's action transpires virtually and immediately. Typical representatives of ROIA are multiplayer online computer games, simulation-based e-learning, and serious gaming. Due to a large, variable number of users, with intensive and dynamic interactions, ROIA make high Quality of Service (QoS) demands on the underlying network. Furthermore, these demands may continuously change, depending on the number of users and the actual application state: in a shooter game, a high packet loss in a combat state may have fatal consequences on QoS, whereas it is less relevant when a player is exploring the terrain.

ROIA Applications has two parts, a static and a dynamic part. Static part has non-changeable and landscape objects. The other one dynamic part has non-playing characters controlled by server. These objects can change their state at any time. Figure 1 shows the structure of a ROIA.

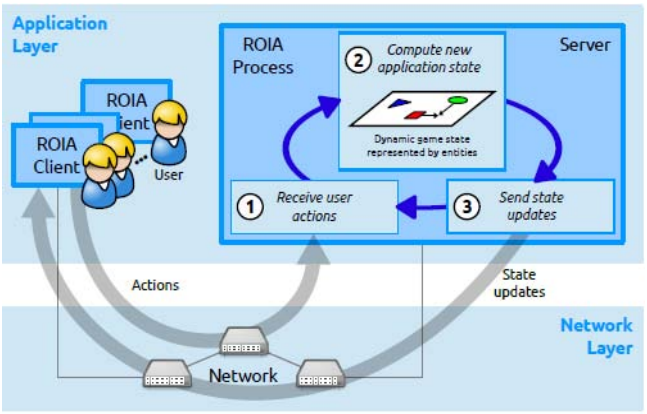

Figure 1: Structure of a ROIA and its real-time loop

In this architecture only one ROIA process serves the connected ROIA clients and a group of ROIA processes distributes among several machines. For the processing of ROIA in an infinite loop the application state executes repeatedly in real time which is called real-time loop [23]. Single loop iteration has three important steps. Firstly, the user takes input and transmits it via a network and ROIA process receives that. Then to calculate the application state we can apply the user input and processing methods to current

application state. After that, the loop transfers the updated state to the client.

2. Multiple Packet Scheduler

The Open Flow data path plus QoS modules form the QoSFlow datapath. This path is a utilizer space implementation and locates in the kernel space. The QoS module opens a channel with the kernel through Netlink and Packet socket families to connect both utilizer and kernel space. Thus, the packet schedulers can be instantiated to enable traffic shaping and enqueueing of flows. Figure 2 shows the components like Traffic Shaping, Packet Schedulers and enqueueing that constructs the QoS module of the QoSFlow datapath, and their relationships.

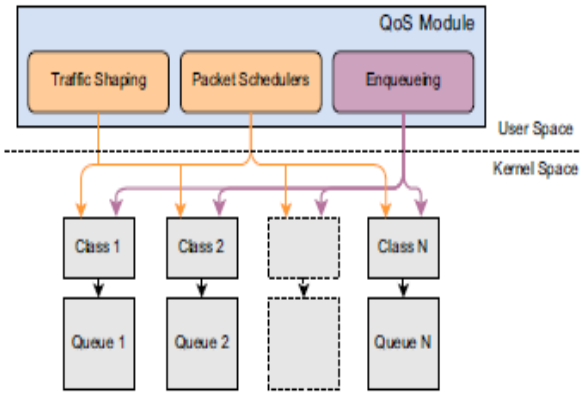

Figure 2: QoS module which has been added to the standard OpenFlow datapath

- *Traffic Shaping and Packet Schedulers:* These components use Netlink socket family to manipulate OFPT_QOS_QUEUEING_DISCIPLINE message type, which is a new extension of the message to represent the QoS messages 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 connect the kernel, these components open a Netlink socket channel and send a Netlink message through it. The Netlink message is the type of message that Linux kernel accepts for network resources management. In this way, the QoS messages maps to Netlink messages.

- *Enqueueing:* It is the component responsible for operating 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 skb-> priority of kernel's data structure called sk_buff. This configuration establish through the use of the SO_PRIORITY option of the Packet socket family. Since user space cannot access such data structure directly.

The QoS development strategy for OF enabling networks is to overcome the packet scheduling issues. The primary goal of QoSFlow is to allow control of multiple packet schedulers. In another word, QoSFlow brings the traffic control of Linux to become part of OF networks. Our proposal extends the OF protocol 1.0 and the standard data path based on it. This way, developers can deploy their applications, for instance, a control of bandwidth according to need with one or more packet schedulers on the network. Currently, QoSFlow provides control of the following packet schedulers: HTB (Hierarchical Token Bucket) [24], RED (Randomly Early Detection) [25], and SFQ (Stochastic Fairness Queuing) [26].

Currently, QoSFlow 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 QoSFlow features come from these Linux kernel packet schedulers.

- *HTB:* It allows splitting the bandwidth according to the 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 the physical link.
- *SFQ:* This belongs to fair queuing algorithm. The SFQ schedules the packets 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 packet scheduler has a threshold value to mark packets to be discarded after queue length becomes greater than the threshold value.

3. NOX

Network operating system (NOX) enables management applications to be constructed as centralized programs over high-level abstractions of network resources as an inverse to the distributed algorithms over low-level addresses [22, 23]. The network operating system does not manage the network itself; it provides a programming interface with high-level abstractions of network resources (e.g., memory, disk storage volume, CPU processing power, disk storage volume, link capacity, etc.) that enable network application programs to perform complicated tasks safely and efficiently on a broad heterogeneity of networking technologies [22]. The NOX, however, fails in giving the necessary functions for QoS-guaranteed Software Defined Networking (SDN) [24] service provisioning on bearer grade provider Internet, such as

QoS-aware virtual network seating, end-to-end network QoS measurement, and cooperation among control elements with other domain network.

IV. Comparison Among the Existing Architectures

1. ROIA

a. *The Response time of ROIA*

Figure 3 shows the graph of the calculation of Response Time with ROIA [1] of Table 1.

Table 1: Calculation of the Response Time with ROIA

Number of Clients	Response Time (ROIA) ms
5	1.03
10	1.19
15	1.22
20	1.35
25	1.29
30	1.07
35	1.48
40	1.21
45	1.34
50	1.09
55	1.42
60	1.3
65	1.15
70	1.45

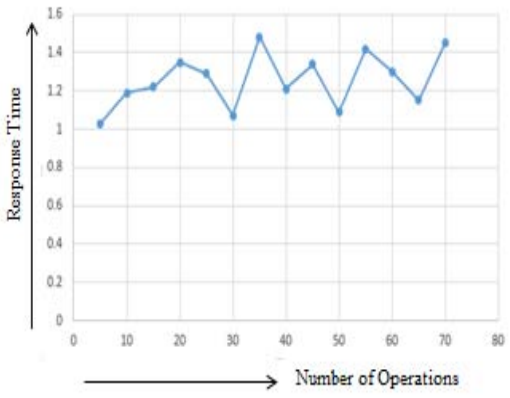

Figure 3: Response Time of ROIA

b. *Throughput of ROIA*

The Figure 4 shows the graph of the calculation of Throughput with ROIA [1] of Table 2.

Table 2: Calculation of the Throughput with ROIA

Number of Clients	Throughput (ROIA) in milliseconds (ms)
5	0.97087
10	0.84033
15	0.81967
20	0.74074
25	0.77519
30	0.93457
35	0.67567
40	0.82644
45	0.74626
50	0.91743
55	0.70422
60	0.76923
65	0.86959
70	0.68965

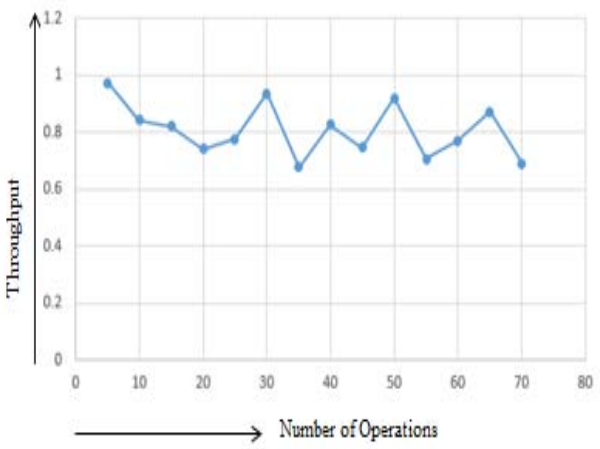

Figure 4: Throughput of ROIA

2. Multiple Packet Scheduler

a. *The Response time of Multiple Packet Scheduler*

Figure 5 shows the graph of the calculation of Response Time with Multiple Packet Schedulers [2] of Table 3.

2. Multiple Packet Scheduler

a. *The Response time of Multiple Packet Scheduler*

Figure 5 shows the graph of the calculation of Response Time with Multiple Packet Schedulers [2] of Table 3.

Table 3: Calculation of the Response Time with Multiple Packet Schedulers

Number of Clients	Response Time (HTB) ms	Response Time (SFQ) ms	Response Time (RED) ms
5	1.28	0.1	0.55
10	2.56	0.2	1.1
15	3.84	0.3	1.65
20	5.12	0.4	2.2
25	6.4	0.5	2.75
30	7.68	0.6	3.3
35	8.96	0.7	3.85
40	10.24	0.8	4.4
45	11.52	0.9	4.95
50	12.8	1	5.5
55	14.08	1.1	6.05
60	15.36	1.2	6.6
65	16.64	1.3	7.15
70	17.92	1.4	7.7

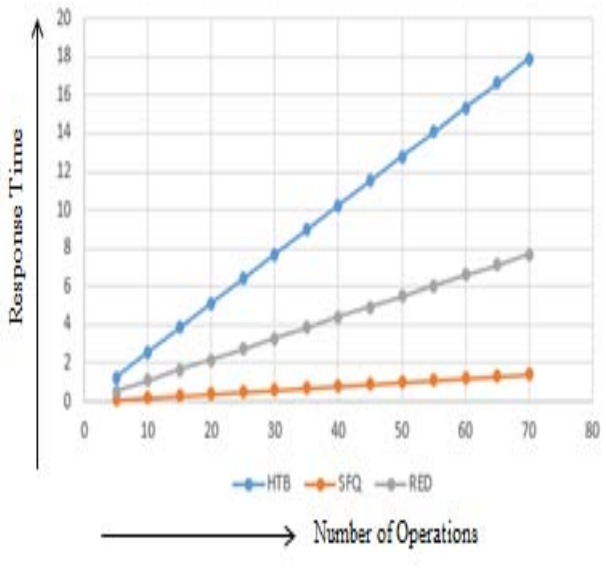

Figure 5: Response Time of Multiple Packet Schedulers

b. *The Throughput of Multiple Packet Scheduler*

Figure 6 shows the graph of the calculation of Throughput with Multiple Packet Schedulers [2] of Table 4.

Table 4: Calculation of the Throughput with Multiple Packet Schedulers

Number of Clients	Throughput (HTB) ms	Throughput (SFQ) ms	Throughput (RED) ms
5	0.12206	0.15625	0.0284
10	0.244125	0.3125	0.05681
15	0.36618	0.46875	0.08522
20	0.48825	0.625	0.113635
25	0.6103125	0.78125	0.14204
30	0.732375	0.9375	0.17045
35	0.8544375	1.09375	0.19886
40	0.9765	1.25	0.22727
45	1.09856	1.40625	0.25567
50	1.220625	1.5625	0.28408
55	1.34268	1.71875	0.31249
60	1.46472	1.875	0.340905
65	1.58678	2.03125	0.36931
70	1.70884.	2.1875	0.39772

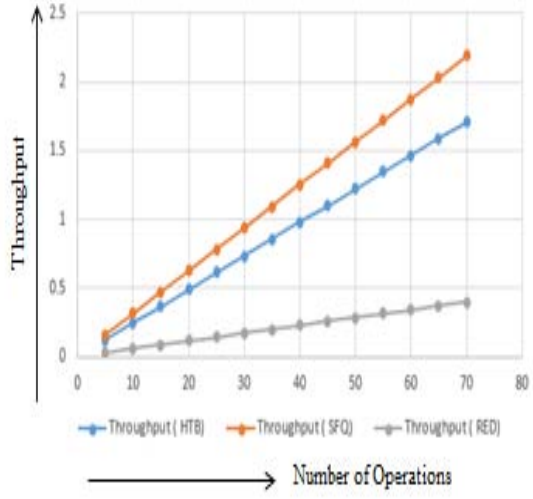

Figure 6: Throughput of Multiple Packet Schedulers

3. NOX

a. *The Response time of NOX*

Figure 7 shows the graph of the calculation of Response Time with NOX [27] of Table 5.

Table 5: Calculation of the Response Time with NOX

Number of Clients	Response Time (NOX) ms
5	0.7948
10	0.983
15	0.8542
20	0.808
25	0.888
30	0.899
35	0.9585
40	0.97
45	0.787
50	0.9095
55	0.755
60	0.7777
65	0.842
70	0.7888

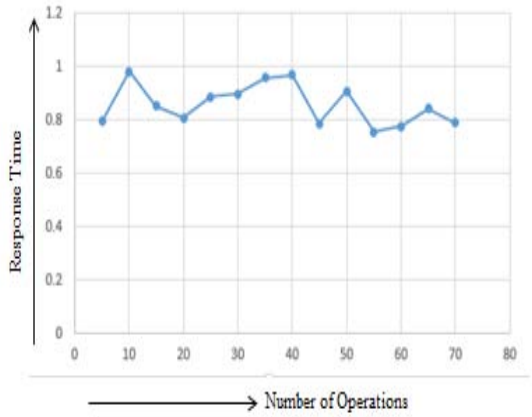

Figure 7: Response Time of NOX

b. *Throughput of NOX*

Figure 8 shows the graph of the calculation of Throughput with NOX [27] of Table 6.

Table 6: Calculation of the Throughput with NOX

Number of Clients	Throughput (NOX) ms
5	0.09
10	0.091
15	0.08
20	0.075
25	0.0859
30	0.0848
35	0.079
40	0.082
45	0.072
50	0.0797
55	0.0976
60	0.0776
65	0.0923
70	0.0948

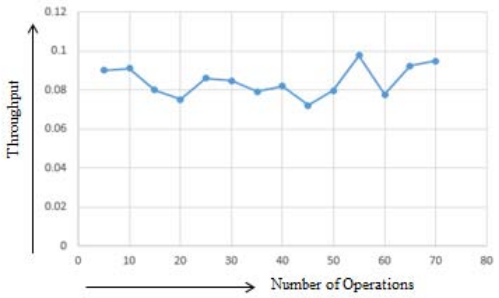

Figure 8: Throughput of NOX

V. Comparison among ROIA, Multiple Packet Scheduler, and NOX

a) Response Time

The response time of HTB is better than the response time of ROIA, NOX, SFQ and RED packet scheduler shown in figure 9.

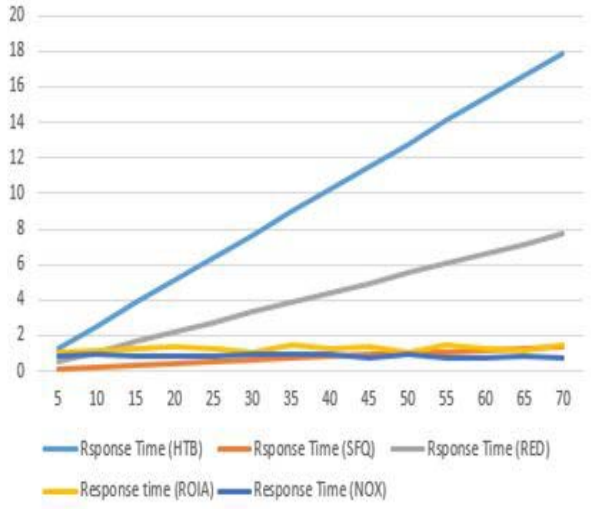

Figure 9: Response time of Existing Architectures

b) Throughput

The throughput of ROIA is better than of HTB, NOX, SFQ and RED packet scheduler in the transmission of first 24 packets, for other 46 packets Throughput of SFQ packet scheduler is better than ROIA, NOX, RED and SFQ packet scheduler shown in figure 10.

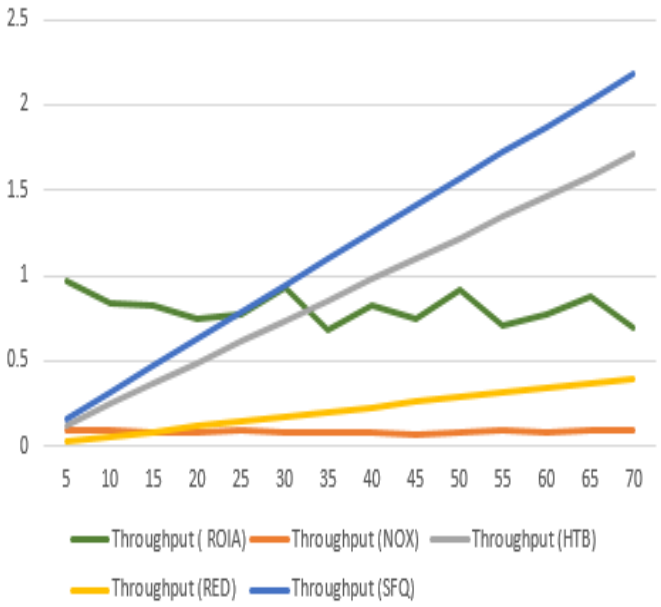

Figure 10: The throughput of Existing Architectures

VI. Conclusion and Future Work

Software Defined Network 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 is the major target for this work. Not only this but also ensuring the security is another purpose for this work. Software Defined Networking (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, OpenFlow claims to provide a standardized interface, the existing diversity of OpenFlow enabled switches leads to different behavior for the same QoS mechanisms. As compared to the response time of existing architectures HTB packet scheduler is better. In case of throughput, SFQ packet scheduler is better. We will improve Quality of Service (QoS) in SDN by designing an efficient architecture and implement that in any network emulator. In the future, we will work with Switch Capacity, Number of Queues Impact, QoE Evaluation, and Bandwidth Isolation.

References Références Referencias

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. Natasha Gude et al., "NOX: Towards an Operating System for Networks," editorial note submitted to CCR.
4. ArsalanTavakoli et al, "Applying NOX to the Datacenter," in Proc. Of SIGCOMM Hotnet 2009.
5. DimitriStaessens et al., "Software Defined Networking: Meeting Carrier Grade Requirements," in Proc. of IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), 2011.
6. 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.
7. 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.
8. A 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. A 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.
15. 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.
16. Hong, C.Y., et al.: Achieving high utilization with software-driven WAN. In: Proceedings of the ACM SIGCOMM, Hong Kong, China, pp. 15–26 (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 data centers via a bargaining game approach. In: Proceedings ofthe 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. ACMSIGCOMM 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. Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. Networking, IEEE/ACM Transactions on, 1(4): 397–413, 1993.
26. 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, pages 733–740. IEEE, 1990.
27. “On Scalability of Software-Defined Networking - IEEE Xplore Document.” 14 February 2013.