Global Journal of Computer Science and Technology: E Network, Web & Security
Volume 19 Issue 1 Version 1.0 Year 2019
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

SDN-Based Approach to Evaluate the Best Controller: Internal Controller NOX and External Controllers POX, ONOS, RYU

By Mohammad Nowsin Amin Sheikh, Monishanker Halder, Sk. Shalauddin Kabir, Md. Wasim Miah & Sawrnali Khatun

Jessore University of Science and Technology

Abstract- Software Defined Networking (SDN) is a rising technique to deal with replace patrimony network (coupled hardware and software program) control and administration by separating the control plane (software program) from the information plane (hardware). It gives adaptability to the engineers by influencing the focal control to plane straightforwardly programmable. Some new difficulties, for example, single purpose of disappointment, may be experienced because of the original control plane. SDN concentrated on flexibility where the security of the system was not essentially considered. It promises to give a potential method to present Quality of Service (QoS) ideas in the present correspondence networks. SDN automatically changes the behavior and functionality of system devices utilizing a single state program. Its immediate OpenFlow is planned by these properties. The affirmation of Quality of Service (QoS) thoughts winds up possible in a versatile and dynamic path with SDN. It gives a couple of favorable circumstances including, organization and framework versatility, improved exercises and tip-top performances.

Keywords: *SDN, QoS, NOX, POX, ONOS, CPU, TCP, OS, BSD, WAN, ODL, FIB, IP, MAC, API, IT, IOT, IOE.*

GJCST-E Classification: *B.4.2*

Strictly as per the compliance and regulations of:

SDN-Based Approach to Evaluate the Best Controller: Internal Controller NOX and External Controllers POX, ONOS, RYU

Mohammad Nowsin Amin Sheikh [α], Monishanker Halder [σ], Sk. Shalauddin Kabir [ρ], Md. Wasim Miah [ω] & Sawrnali Khatun [¥]

Abstract- Software Defined Networking (SDN) is a rising technique to deal with replace patrimony network (coupled hardware and software program) control and administration by separating the control plane (software program) from the information plane (hardware). It gives adaptability to the engineers by influencing the focal control to plane straightforwardly programmable. Some new difficulties, for example, single purpose of disappointment, may be experienced because of the original control plane. SDN concentrated on flexibility where the security of the system was not essentially considered. It promises to give a potential method to present Quality of Service (QoS) ideas in the present correspondence networks. SDN automatically changes the behavior and functionality of system devices utilizing a single state program. Its immediate OpenFlow is planned by these properties. The affirmation of Quality of Service (QoS) thoughts winds up possible in a versatile and dynamic path with SDN. It gives a couple of favorable circumstances including, organization and framework versatility, improved exercises and tip-top performances. This research work will concentrate on the Quality of Service (QoS) like delay, response time, throughput, and other execution assessing parameters of our proposed arrange design using internal controller, e.g., Network Operating System (NOX) and external controller, e.g., Pythonic Network Operating System (POX), Open Network Operating System (ONOS) and RYU. Regardless of the way that thoughts of QoS, they did not comprehend the correspondence systems with high utilization, diverse quality and acknowledgment costs. It will focus on the outside controller and inner controller execution in the proposed architecture. These perceptions of switch diversity may give SDN application engineer's bits of knowledge while acknowledging QoS ideas in an SDN-based system.

Keywords: SDN, QoS, NOX, POX, ONOS, CPU, TCP, OS, BSD, WAN, ODL, FIB, IP, MAC, API, IT, IOT, IOE.

Author α: Assistant Professor in the department of Computer Science and Engineering in Jessore University of Science and Technology, Bangladesh. e-mail: nowsin.jstu@gmail.com

Author σ: Lecturer in the department of Computer Science and Engineering in Jessore University of Science and Technology, Bangladesh. e-mail: monicsejust@gmail.com

Author ρ: M.Sc. degree in the department of Computer Science and Engineering in Jessore University of Science and Technology, Bangladesh. e-mail: riponcse32@gmail.com

Author ωY¥: B.Sc. degree in the department of Computer Science and Engineering in Jessore University of Science and Technology, Bangladesh. e-mails: wasimcse767@gmail.com, sawrnalikhatun@gmail.com

I. Introduction

Software Defined Networking (SDN) is the current worldview of the systems administration which is taken under consideration by stakeholders with a huge concern. The idea works towards significantly decreasing system organization and administration costs. SDN's innovation is a novel method to manage conveyed figuring that encouraged to arrange administration and empowers system setup automatically, keeping in mind the end goal to enhance monitoring and network performance [1]. It is intended to address the way that the static engineering of the conventional network is decentralized and complex while current systems require greater adaptability and simple investigating. SDN recommends incorporating system insight in one system segment by disassociating the sending procedure of network parcels (Data Plane) from the routing procedure (Control plane). The control plane comprises of at least one controller which is the cerebrum of SDN and the entire insight consolidates there. In any case, the knowledge centralization has its particular downsides with regards to security, [2] versatility and elasticity [3] of SDN. Software Defined Networking [4], [5] is a network system that offers a plan to change the impediments of current system frameworks. In the first place, it breaks the vertical coordination by isolating the system's control logic (the control plane) from the hidden switches and routers that forward the activity (the information plane). Second, with the detachment of the control and information planes, arrange switches end up straightforward sending gadgets and the controlling rationale is executed in a sensibly brought together controller (or system working framework), disentangling strategy implementation and network reconfiguration and advancement [6]. It encompasses different sorts of network headway proposed to make more versatile and spry to help the capacity structure and virtual server of the line server ranches. We will without a lot of an understand SDN against standard framework by a prime portrayal; acknowledged inside the groups that we have to pass on a bundle in the standard; it must change its course thing times for finding the ideal way. It is a proficient structure to use better Quality of Service (QoS) which

exhibits to a framework's profitability to achieve most vital transmission limit and oversee other framework execution parts like inactivity, error rate and run time [7]. An SDN controller is an application in SDN that supervises stream control to engage smart frameworks administration. SDN controllers rely upon convention, for instance, OpenFlow, that empower servers to encourage changes where to send Packets.

Network Operating Systems (NOX) is the first OpenFlow controller. It fills in as a system control stage that gives an abnormal state automatic interface for administration and the advancement of system control applications. Its framework reflects the change and sorting out into a product issue. NOX is a bit of the software-defined networking (SDN) biological system. In particular, it's a stage for building system control applications. The main SDN innovation to get acknowledgment is OpenFlow, and NOX is the first creation at Nicira Networks next to each other with OpenFlow — NOX was the primary OpenFlow controller. Nicira gave NOX to the examination group in 2008, and from that point forward, it has been the reason for some and different research extends in the early investigation of the SDN space. Its applications are different kinds of activities on an abnormal state of nonattendance of unfaltering quality in organizing execute fragment, not at all like bring down back of estimation game-plan [8][9] applications. The system working structure does not manage itself; It outfits a programming interface with top state objects, (for instance, plate accumulating volume, CPU preparing power, memory control, etc.) of framework resources, which engages sort out application undertakings to manage secure and down to earth complex assignments on a combination of network [9]. The NOX, in any case, fails in giving the required capacities to QoS-guaranteed Software Defined Networking (SDN) [10] convenience provisioning on transporter review supplier Internet, for example, QoS-cautious virtual network inserting, end-to-end arrange QoS appraisal, and joint efforts among control components for others space network.

Pythonic Network Operating System (POX) [11] is an open source advancement stage for Python-based software-defined networking (SDN) control applications, for example, OpenFlow SDN controllers. The controller gives an effective method to actualize the OpenFlow convention which is the accepted correspondence convention between the controllers and the switches. Utilizing POX controller can run distinctive applications like center point, switch, firewall, and load balancer. TCP dump bundles catch the instruments to see the bundles streaming between POX controller and OpenFlow devices. It uses as a part of SDN network systems since it has a Python dialect interface for research [12]. OpenFlow, when all says in done has pulled in considerable enthusiasm from industry [13] [14]. POX formally requires Python 2.7 (however quite a bit of it will work fine with Python 2.6) and should keep running under Linux, Mac OS, and Windows. (What's more, pretty much anyplace else - we've run it on Android telephones, under FreeBSD, Haiku, and somewhere else).

Open Network Operating System (ONOS) [15] is the main open source SDN controller for building next-generation SDN/NFV solutions. It gives the control plane to a software-defined network (SDN), overseeing network segments, for example, switches and connections, and running programming projects or modules to give correspondence administrations to end has and neighboring systems. ONOS stages do flaunt being intended to help, like different controllers, different application classifications, for example, control, setup and the executive's applications. Among the applications that are distributed by ON.Lab, some of them are Segment Routing, multi-layer SDN control, topology watcher, way calculation and SDN-IP peering applications.[16][17] ONOS strengthen different types of southbound protocols like OpenFlow, NetConf, and so on., for correspondence with an assortment of net gadgets. ONOS, like different controllers (ex: ODL), utilizes the idea of suppliers — one each for each southbound convention – which conceals convention unpredictability from different segments of the controller stage. These suppliers give all the essential 'depictions' of system components profoundly layer. [18]

RYU Controller is an open; Software Defined Networking (SDN) Controller intended to expand the deftness of the system by making it simple to oversee and adjust how traffic is taken carefully. By and large, the SDN Controller is the brains of the SDN condition, imparting data down to the switches and switches with southbound APIs, and up to the applications and business rationale with northbound APIs. It gives programming segments, with all around characterized application program interfaces (APIs) that make it simple for designers to make new system the executives and control applications. This segmented approach encourages associations to redo arrangements to meet their particular needs; engineers can rapidly and effortlessly adjust existing segments or execute their very own to guarantee the fundamental system can meet the changing requests of their applications. RYU bolsters different conventions for overseeing system gadgets, for example, OpenFlow, NetConf, OF-Config, and so forth. [19]

II. Related Work

Past work on giving QoS ensures utilizing Open-Flow can be divided into three classifications. To start with, studies about conveying dynamic QoS in an SDN domain [20][21][22]. Second, ponders on switch assorted variety [23] [24] [25] [26].Third, look into on

arranging execution coming about because of QoS with OpenFlow-empowered switches [27], [28].

A portion of the work done in the region of SDN-based, on request provisioning of system assets is concentrated towards computerized, strategy based system provisioning [29] [30], while other focuses towards movement designing crosswise over Wide Area Networks (WANs) [31][32]. Dynamic distribution of system assets additionally requires inside the server farms, and numerous Investigations address this test. For instance, an OpenFlow-based calculation for designation of transfer speed assets between virtual machines in server farms is exhibited in [33], while in [34] the creators depict a stage for incorporated provisioning of a process, stockpiling and system assets in server farms.

Lately, different specialists have been done to break down SDN controllers like NOX, ODL, ONOS, and RYU and so on. Other research paper had found out the beginning consequences of benchmarking through Open-Daylight SDN external Controller with Floodlight controller. The Researcher had assessed throughput, inaction and response time of Open Daylight SDN Controller and Floodlight under various conditions [35]. A related undertaking of particular note is Maestro [36] (made in parallel to NOX), which also charge as a "framework working system." Through NOX controller 4D structures are to control sending (e.g., FIBs in switches), and in this way their frameworks consolidate to arrange an establishment (e.g., joins, switches/switches). The Rational [37] and Ethane [38] wanders give more broad class of convenience by including a namespace for customers and center points in their framework view and observing the ties between these names and the low-level IP addresses and MAC.

Scientists were Used POX Controller for measuring the Network Programmability [39]. POX is used to examine the Performance parameters including inactivity and throughput, service quality, versatility, delay, response time and security. The exploration of execution and adaptability finish with Clench. Security and service quality test has alter with probe.

III. Description of the Proposed Architecture

To measure the Quality of Service (QoS) like throughput, response time, delay and other execution evaluating parameters, we made SDN based cloud architecture. In our created engineering, there are nine routers connected with three cloud interfaces. These various routers connect with eight hosts which have an IP address. For measuring QoS shows from hosts to switches, SDN controllers use as existing controllers and external controllers.

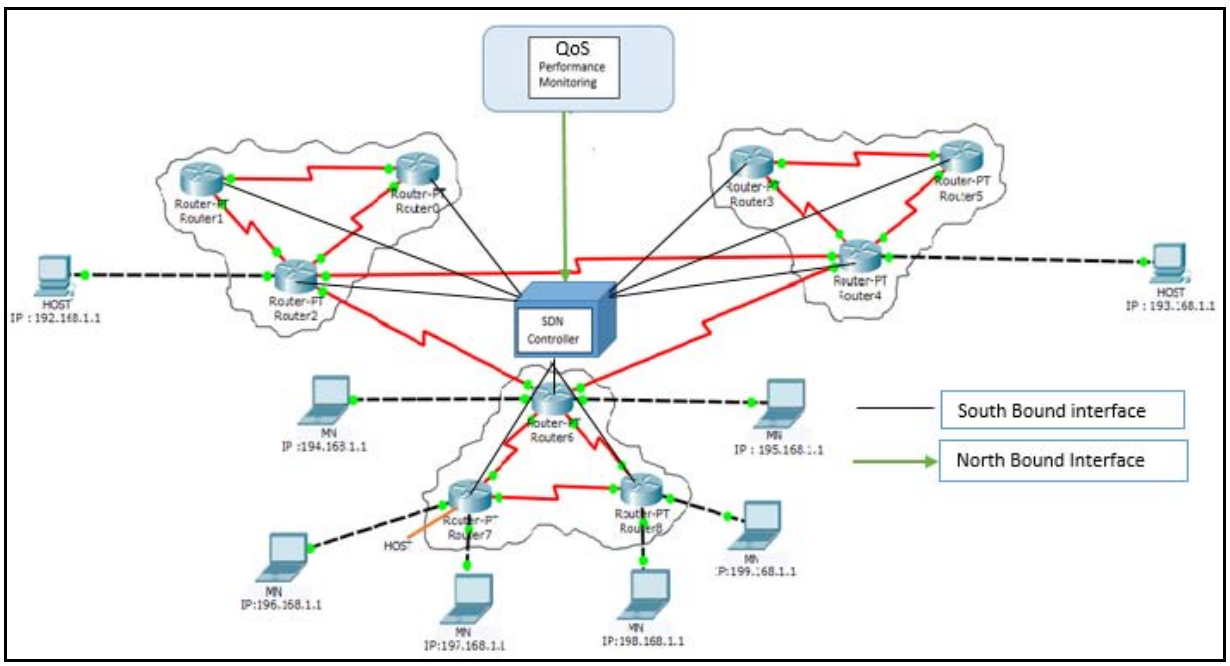

Fig. 1: Proposed_SDN_architecture and its real-time loop

a) Architectural description based on NOX controller

Network operating system (NOX) empowers administration applications to develop as brought together projects over high state reflections of system assets as an inverse to the dispersed calculations over low-level locations [40, 41]. The essential parts of a NOX-based system: an arrangement of switches and at least one organize appended servers. The NOX programming (and the administration applications that keep running on NOX) keep running on these servers. IT

includes a few distinctive controller forms (commonly one on each system appended server), and a system sees (this keep in a database running on one of the servers). The arrange see contains the consequences of NOX's system perceptions; applications utilize this state to settle on administration choices. For NOX to control organized the activity, it must control arrange switches; for this reason, we use the switches that help the OpenFlow switch reflection [42].

b) Architectural description based on POX controller

POX, which creates in python, is a tip-top, open source OpenFlow controller. An SDN controller developer develops POX in light of Haiku, a trial OpenFlow controller from Stanford University. Huge Switch Networks backs POX as an association that essentially offers answers for business data centers. It offers different features and consultations for controlling an OpenFlow organize. For perfect utilization of benefits, POX relies upon multi-threading and can manage a couple of million new streams for each second. The Westbound python allows the change of custom modules in python and quick interfacing with the middle controller. The modules are stacked by methods for an alternate structure when the POX controller starts. You would along these lines have the capacity to utilize the full handiness of the controller and OpenFlow API and speedily respond to events on the framework, for instance, the ascent of new packages or new streams. The Open Flow datapath despite QoS modules shapes the QoS Flow datapath. This datapath is a consumption space utilize where lines mastermind in the portion space. The QoS module develops a station with the bit through Netlink to relate for both service and bit map. Like this, the POX can instantiated to connect with activity adornment and enqueuing of streams in our proposed engineering.

c) Architectural description based on ONOS controller

ONOS was intended to address the issues of administrators wishing to construct bearer level arrangements that use the financial aspects of white box vendor silicon equipment while offering the adaptability to make and send new unique system administrations with disentangled automatic interfaces. It underpins both setup and constant control of the system, disposing of the need to run directing and exchanging control conventions inside the system texture. By moving knowledge into the ONOS cloud controller, advancement empowers, and end-clients can undoubtedly make new system applications without the need to adjust the data-plane frameworks. In ONOS, in contrast to different controllers, disseminated engineering support is one of the plan standards and not an idea in retrospect bolster. It is additionally like five to six of the dispersed models portrayed above in the Distributed SDN Controller Architecture segment. That is, ONOS can be sent as the gathering of controller-servers that facilitate with one another to accomplish strength, adaptation to non-critical failure, and better load the executives.

d) Architectural description based on RYU controller

RYU controller is single-strung substances which execute different functionalities in RYU. Events are messages between them. It sends offbeat occasions to one another. Other than that, there are some RYU-inner occasion sources which are not RYU applications. One of the instances of such occasion sources is Open-stream controller. While an occasion can as of now contain self-assertive python protests, it's debilitated to pass complex items (e.g. unpick capable terms) between RYU. Each RYU controller has a get line for occasions. It has a string for occasion preparing. The RYU controller continues depleting the get line by lining a datapath and calling the proper occasion handler for the occasion's type. Since the occasion handlers bring with regards to the occasion preparing string, it ought to be cautious when blocking. While an occasion handler hinders, no further occasions for the RYU application will process. There are sorts of handlers which are utilized to actualize synchronous between application calls between RYU controller.

IV. Comparison Between the Internal and External Controllers on our Proposed Architecture

a) NOX controller

i. *Response Time of NOX* controller

Figure 2 shows the graph of the calculation of Response Time with NOX of Table 1.

Table 1: Calculation of Response Time with NOX controller

Number of Operations	Response Time (NOX) ms
4	0.028
8	0.029
12	0.030
16	0.029
20	0.031
24	0.032
28	0.033
32	0.032
36	0.033
40	0.031
44	0.034
48	0.033
52	0.034
56	0.034
60	0.035
64	0.034

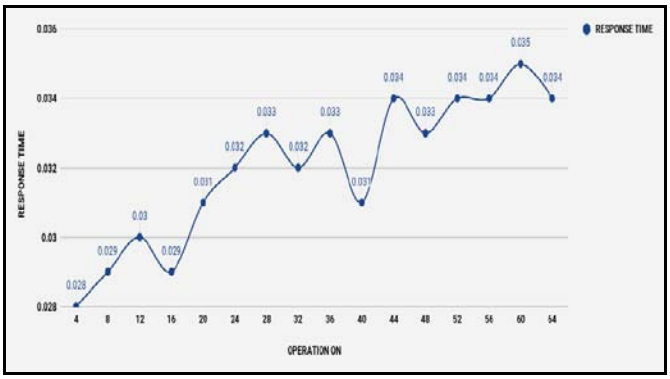

Fig. 2: Response Time of NOX controller

ii. *Throughput of NOX* controller

Figure 3 shows the graph of the calculation of Throughput with NOX of Table 2.

Table 2: Calculation of Throughput with NOX controller

Number of Operations	Throughput (NOX) ms
4	0.138
8	0.156
12	0.234
16	0.274
20	0.279
24	0.274
28	0.231
32	0.234
36	0.275
40	0.274
44	0.187
48	0.234
52	0.244
56	0.217
60	0.217
64	0.187

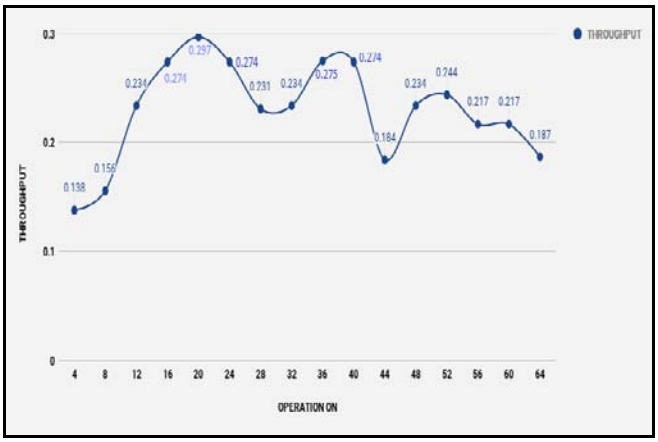

Figure 3: Throughput of NOX controller

b) Pythonic Network Operating System (POX) controller

i. *Response Time of POX controller*

Figure 4 represents the graph for the calculation of Response Time with POX of Table 3.

Table 3: Calculation of Response Time with POX controller

Number of Operations	Response Time (POX) ms
4	0.045
8	0.041
12	0.037
16	0.039
20	0.039
24	0.040
28	0.039
32	0.038
36	0.037
40	0.036
44	0.036
48	0.041
52	0.040
56	0.044
60	0.043
64	0.042

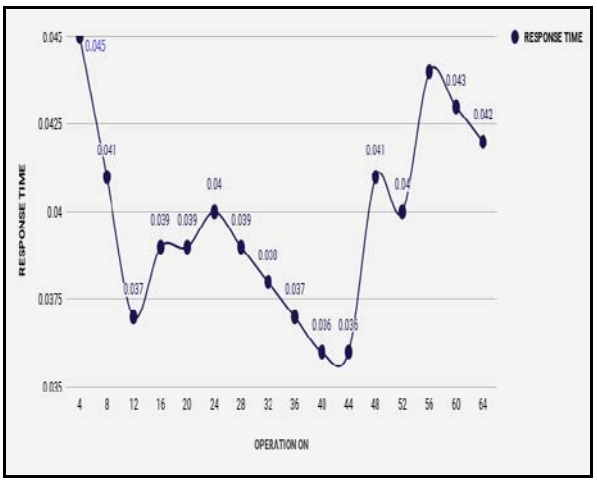

Figure 4: Response Time of POX controller

ii. *Throughput of POX controller*

Figure 5 shows the graph of the calculation of Throughput with POX of Table 4.

Table 4: Calculation of Throughput with POX controller

Number of Operations	Throughput (POX) ms
4	0.660
8	0.751
12	0.692
16	0.745
20	0.649
24	0.673
28	0.604
32	0.589
36	0.554
40	0.533

44	0.510
48	0.502
52	0.476
56	0.462
60	0.442
64	0.420

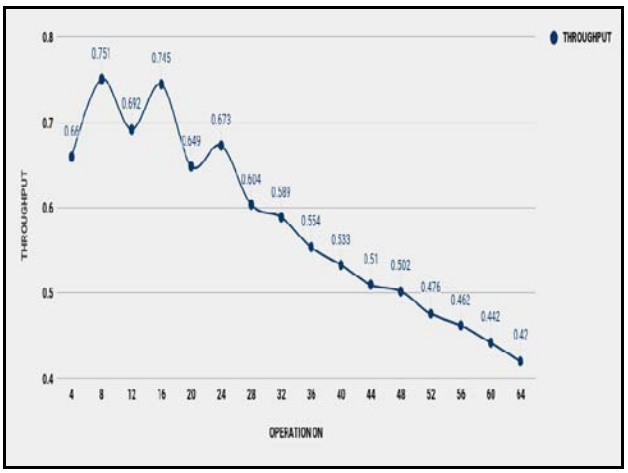

Figure 5: Throughput of POX controller

c) *Open Network Operating System (ONOS) controller*

i. *Response Time of ONOS controller*

Figure 6 shows the graph of the calculation of Response Time with ONOS of Table 5.

Table 5: Calculation of Response Time with ONOS controller

Number of Operations	Response Time (ONOS) ms
4	0.070
8	0.063
12	0.060
16	0.058
20	0.057
24	0.056
28	0.055
32	0.054
36	0.053
40	0.052
44	0.050
48	0.049
52	0.049
56	0.047
60	0.047
64	0.048

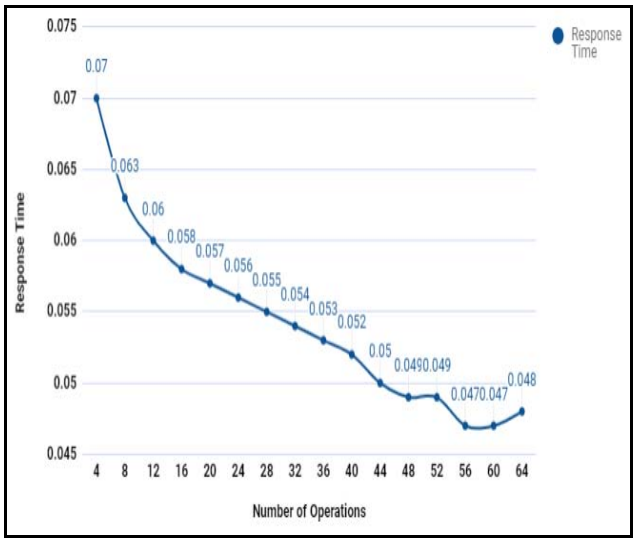

Figure 6: Response Time of ONOS controller

ii. *Throughput of ONOS controller*

Figure 7 shows the graph of the calculation of Throughput with ONOS of Table 6.

Table 6: Calculation of Throughput with ONOS controller

Number of Operations	Throughput (ONOS) ms
4	0.634
8	0.449
12	0.351
16	0.303
20	0.228
24	0.180
28	0.161
32	0.122
36	0.121
40	0.118
44	0.117
48	0.129
52	0.116
56	0.123
60	0.119
64	0.127

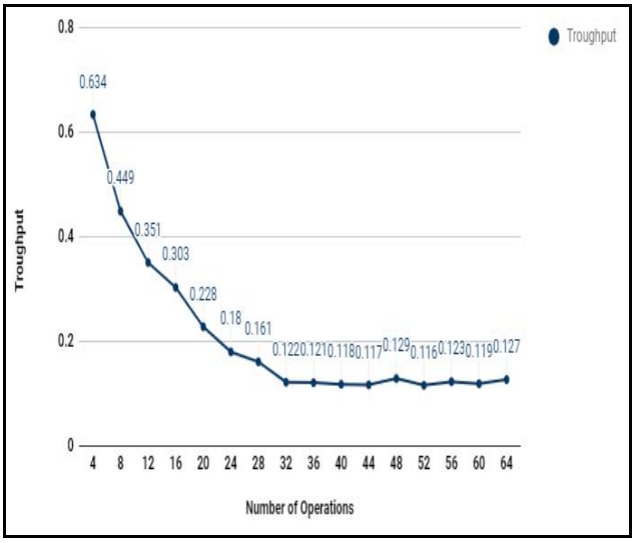

Figure 7: Throughput of ONOS controller

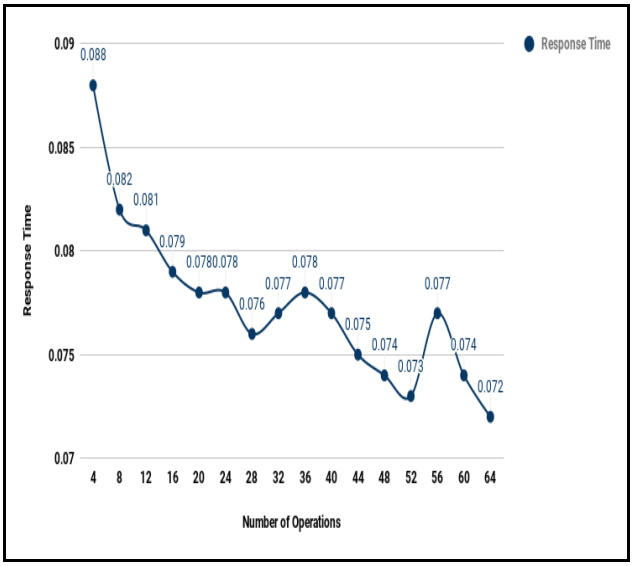

Figure 8: Response Time of RYU controller

d) RYU controller

i. *Response Time of RYU controller*

Figure 8 shows the graph of the calculation of Response Time with RYU of Table 7.

Table 7: Calculation of Response Time with RYU controller

Number of Operations	Response Time (RYU) ms
4	0.088
8	0.082
12	0.081
16	0.079
20	0.078
24	0.078
28	0.076
32	0.077
36	0.078
40	0.077
44	0.075
48	0.074
52	0.073
56	0.077
60	0.074
64	0.072

ii. *Throughput of RYU controller*

Figure 9 shows the graph of the calculation of Throughput with RYU of Table 8.

Table 8: Calculation of Throughput with RYU controller

Number of Operations	Throughput (RYU) ms
4	1.177
8	0.652
12	0.442
16	0.460
20	0.251
24	0.435
28	0.251
32	0.215
36	0.435
40	0.188
44	0.262
48	0.271
52	0.195
56	0.276
60	0.269
64	0.178

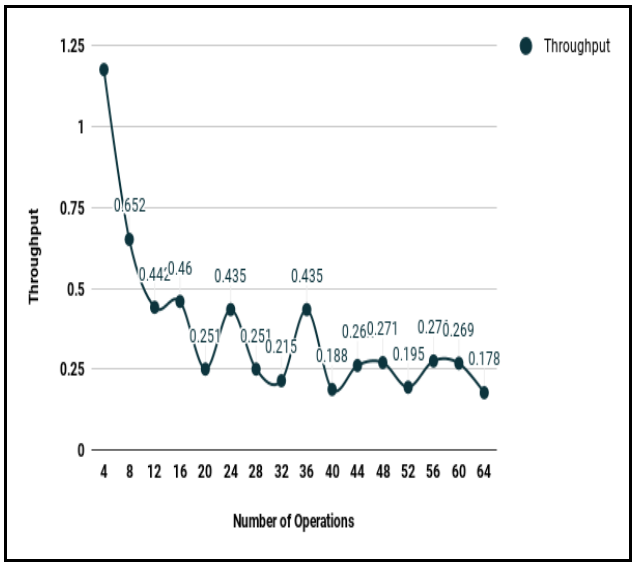

Figure 9: Throughput of RYU controller

V. Performance Analysis Equation

We executed 64 operations in our Proposed_SDN_architecture. In the comparison of response time and throughput among internal controller Network operating system (NOX) and external controllers Pythonic Network Operating System (POX), Open Network Operating System (ONOS) and RYU.

The performance of response time and throughput in our Proposed_SDN_architecture can be calculated against internal and external controllers by the following equation:

Let, Performance (P) = M/N

Where N = Sum of the response time or the throughput of internal controller NOX.

M = Sum of the response time or the throughput of external controllers POX, ONOS, and RYU.

VI. Performance Analysis Among Internal Controller Nox And External Controllers Pox, Onos, Ryu

a) Comparison of Response Time

i. *Response Time between NOX and POX, NOX and ONOS, NOX and RYU controllers*

The performance of response time in our Proposed_SDN_architecture can be calculated against internal controller NOX and external controllers POX, ONOS and RYU by the following equation:

NOX and POX:

Let, Performance (P) = M/N

Where N = sum of the response time of internal controller NOX

M = sum of the response time of external controller POX

Here, M = 0.637

N = 0.512

So, P = (0.637/0.512) times

= 1.244 times

NOX and ONOS:

Let, Performance (P) = M/N

Where N = sum of the response time of internal controller NOX

M = sum of the response time of external controller ONOS

Here, M = 0.868

N = 0.512

So, P = (0.868/0.512) times

= 1.695 times

NOX and RYU:

Let, Performance (P) = M/N

Where N = sum of the response time of internal controller NOX

M = sum of the response time of external controller RYU

Here, M = 1.239

N = 0.512

So, P = (1.239/0.512) times

= 2.420 times

ii. *Throughput between NOX and POX, NOX and ONOS, NOX and RYU controllers*

The performance of throughput in our Proposed_SDN_architecture can be calculated against internal controller NOX and external controllers POX, ONOS and RYU by the following equation:

NOX and POX:

Let, Performance (P) = M/N

Where N = sum of the throughput of internal controller NOX

M = sum of the throughput of external controller POX

Here, M = 9.262

N = 3.655

So, P = (9.262/3.655) times

= 2.534 times

NOX and ONOS:

Let, Performance (P) = M/N

Where N = sum of the throughput of internal controller NOX

M= sum of the throughput of external controller ONOS

Here, M = 3.798

N = 3.655

So, P = (3.798/3.655) times

= 1.039 times

NOX and RYU:

Let, Performance (P) = M/N

Where N = sum of the throughput of internal controller NOX

M = sum of the throughput of external controller RYU

Here, M = 5.957
N = 3.655
So, P = (5.957/3.655) times
= 1.630 times

VII. Performance Analysis between External Controllers Pox, Onos, RYU

a) Comparison of Response Time

i. *Response Time between POX and ONOS, POX and RYU, ONOS and RYU controllers*

The performance of response time in our Proposed_SDN_architecture can be calculated against external controllers POX, ONOS, and RYU by the following equation:

POX and ONOS:

Let, Performance (P) = M/N

Where N = sum of the response time of external controller POX

M = sum of the response time of external controller ONOS

Here, M = 0.868
N = 0.637
So, P = (0.868/0.637) times
= 1.363 times

POX and RYU:

Let, Performance (P) = M/N

Where N = sum of the response time of external controller POX

M = sum of the response time of external controller RYU

Here, M = 1.239
N = 0.637
So, P = (1.239/0.637) times
= 1.945 times

ONOS and RYU:

Let, Performance (P) = M/N

Where N = sum of the response time of external controller ONOS

M = sum of the response time of external controller RYU

Here, M = 1.239
N = 0.868
So, P = (0.651/0.528) times
= 1.427 times

ii. *Throughput between POX and ONOS, POX and RYU, ONOS and RYU controllers*

The performance of throughput in our Proposed_SDN_architecture can be calculated against external controllers POX, ONOS and RYU by the following equation:

POX and ONOS:

Let, Performance (P) = M/N

Where N = sum of the throughput of external controller ONOS

M = sum of the throughput of external controller POX

Here, M = 9.262
N = 3.798
So, P = (9.262/3.798) times
= 2.439 times

POX and RYU:

Let, Performance (P) = M/N

Where N = sum of the throughput of external controller RYU

M = sum of the throughput of external controller POX

Here, M = 9.262
N = 5.957
So, P = (9.262/5.957) times
= 1.555 times

ONOS and RYU:

Let, Performance (P) = M/N

Where N = sum of the throughput of external controller ONOS

M = sum of the throughput of external controller RYU

Here, M = 5.957
N = 3.798
So, P = (5.957/3.798) times
= 1.568 times

VIII. Result Discussion

In our Proposed_SDN_architecture, we executed 64 operations, and we find the better QoS from the architecture through internal and external SDN controller. After measuring performance, we find that internal SDN controller is better than external SDN controller. As a result of comparison between internal controller NOX and external controllers POX, ONOS and RYU, the following results find-

- The response time of internal controller NOX is **1.244x** higher than the response time of external controller POX.
- The response time of internal controller NOX is **1.695x** higher than the response time of external controller ONOS.
- The response time of internal controller NOX is **2.420x** higher than the response time of external controller RYU.
- So, internal SDN controller **NOX** is better in the performance of response time than other controllers.

- The throughput of internal controller NOX is **2.534x** better than the throughput of external controller POX.
- The throughput of internal controller NOX is **1.039x** better than the throughput of external controller ONOS.
- The throughput of internal controller NOX is **1.630x** better than the throughput of external controller RYU.
- So, internal SDN controller **NOX** is better in the performance of throughput than other controllers.

As a result of comparison between external controllers POX, ONOS, and RYU, the following results find-

- The response time of external controller POX is **1.363x** higher than the response time of external controller ONOS.
- The response time of external controller POX is **1.945x** higher than the response time of external controller RYU.
- The response time of external controller ONOS is **1.427x** higher than the response time of external controller RYU.
- So, external SDN controller **POX** is better in the performance of response time than other external controllers.
- The throughput of external controller ONOS is **2.439x** better than the throughput of external controller POX.
- The throughput of external controller RYU is **1.555x** better than the throughput of external controller POX.
- The throughput of external controller ONOS is **1.568x** better than the throughput of external controller RYU.
- So, external SDN controller **ONOS** is better in the performance of throughput than other external controllers.

IX. Conclusion And Future Work

Software-defined networking (SDN) is a design implying to be dynamic, reasonable, cost-effective, and versatile, trying to be appropriate for the high-data transmission, dynamic nature of the present applications. It enables associations to quicken application sending and conveyance, drastically lessening IT costs through strategy empowered work process automation. SDN technology develops cloud models by giving mechanized, on-request application conveyance and versatility at scale. With SDN the item that harps on these controllers settles on the sum decisions and sends this information down to each physical gadget. The central goal of this work is empowering included regard benefits in sort out system. SDN uses to find the security services in different framework structures. It favors a fundamental and flexible insistence of existing dynamic Quality of Service (QoS) parts in the present correspondence arrange. In the examination of the response time of the proposed architecture, internal controller NOX is superior to external controller POX, ONOS, and RYU. On account of throughput, internal controller NOX is additionally better. Among external controllers, POX is better in the performance of response time, and ONOS is better in the performance of throughput than other external controllers. Using SDN, we will work with Number of Bandwidth Isolation, Queues Impact, QoE Evaluation, and Switch Capacity. We will likewise chip away at stack adjust, security frameworks, remote system, Secure Mobility, Cloud Networking, etc. We will work at IOE or IOT like city automation, industry, home, country etc. through the best Quality of Service (QoS) by adducting our proposed SDN architecture with different controllers.

References Références Referencias

1. Benzekki Kamal et al.Software defined networking (SDN): a survey., Security and Communication Networks 9, no. 18 (2016): 5803-5833.
2. Benzekki Kamal et al.Devolving IEEE 802.1 X authentication capability to data plane in software-defined networking (SDN) architecture. Security and Communication Networks 9.17 (2016): 4369-4377.
3. Benzekki Kamal et al Software-defined networking (SDN): a survey., Security and Communication Networks 9, no. 18 (2016): 5803-5833.
4. N. Mckeown, "How SDN will Shape Networking," October 2011. [Online]. Available: http://www.youtube.com/watch?v=c9-K5O qYgA
5. S. Schenker, "The Future of Networking, and the Past of Protocols," October 2011. [Online]. Available: http://www.youtube.com/watch?v=YHeyuD89n1Y
6. H. Kim and N. Feamster, "Improving network management with software defined networking," Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, 2013.
7. "Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking - IEEE Xplore Document." 12 December 2013.
8. ArsalanTavakoli et al, "Applying NOX to the Datacenter," in Proc. Of SIGCOMM Hotnet 2009.
9. Natasha Gude et al., "NOX: Towards an Operating System for Networks," editorial note submitted to CCR.
10. DimitriStaessens et al., "Software Defined Networking: Meeting Carrier Grade Requirements," in Proc. of IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), 2011.
11. Fernandez, Marcial. "Evaluating OpenFlow controller paradigms." In ICN 2013, The Twelfth

International Conference on Networks, pp. 151-157. 2013.
12. Rodrigues Prete, L. et al, "Simulation in an SDN network scenario using the POX Controller," 2014 IEEE COLCOM Conference, pp.1-6, 4-6 June 2014.
13. Levy, S., "Going with the Flow: Google's Secret Switch to the Next Wave of Networking", Wired, April 17, 2012.
14. Neagle, C. "HP takes giant first step into OpenFlow: HP is announcing its first effort to support OpenFlow standard on its Ethernet switches". Network World, February 2012.
15. ON.Lab white paper "Introducing ONOS — A SDN Network Operating System for Service Providers", 2014.
16. Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. 2014. ONOS: Towards an Open, Distributed SDN OS. In "Proceedings of the Third Workshop on Hot Topics in Software Defined Networking" (HotSDN '14). ACM, New York, NY, USA, 1-6
17. Prajakta Joshi, "Introducing ONOS," Webinar, 2014 http://www.opennetsummit.org/ons-inspire-webinars-onlab-onos-nov11.php
18. ON.Lab white paper, "Driving SDN Adoption in Service Provider Networks", 2014.
19. Mohammad Nowsin Amin Sheikh, Monishanker Halder, SK. Shalauddin Kabir, Rathindra Nath Mohalder "Performance Evaluation on Software Defined Networking through External Controller Floodlight and Internal Controller NOX" International Journal of Scientific and Engineering Research (IJSER) - (ISSN 2229-5518), IJSER Volume 9, Issue 7, July 2018.
20. P. Georgopoulos, Y. Elkhatib, M. Broadbent et al., "Towards networkwide 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. Md. Alam Hossain, Mohammad Nowsin Amin Sheikh, Monishanker Halder, Sujan Biswas & Md. Ariful Islam Arman, "Quality of Service in Software Defined Networking", Global Journal of Computer Science and Technology: ENetwork, Web & Security, Volume 18, Issue 3, Version 1.0, Year 2018, Online ISSN: 0975-4172& Print ISSN: 0975-4350.
28. 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.
29. 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.
30. 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).
31. 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.
32. Guo, J., Fangming, L., Haowen, T., Yingnan, L., Hai, J., John, L.: Falloc: fair networkbandwidth allocation in IaaS datacenters via a bargaining game approach. In: Proceedings ofthe ICNP, Gotingen, Germany, pp. 1–10, October 2013.

33. 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)
34. Jain, S., et al.: B4: Experience with a globally-deployed software defined WAN. ACMSIGCOMM Comput. Commun. Rev. 43(4), 3–14 (2013).
35. Z. K. Khattak, M. Awais and A. Iqbal, "Performance evaluation of OpenDaylight SDN controller," 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, Taiwan, 2014, pp. 671-676.
36. Z. Cai, F. Dinu, J. Zheng, A. L. Cox, and T. S. E. Ng. Maestro: A Clean-Slate System for Orchestrating Network Control Components. Under submission, 2008.
37. M. Casado, T. Garfinkel, M. Freedman, A. Akella, D. Boneh, N. McKeown, and S. Shenker. SANE: A Protection Architecture for Enterprise Networks. In Usenix Security Symposium, 2006.
38. M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In SIGCOMM '07, 2007.
39. Shalimov, Alexander, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Smeliansky. "Advanced study of SDN/OpenFlow controllers." In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, p. 1. ACM, 2013.
40. M. Betts, S. Fratini, N. Davis, R. Dolin and others, "SDN Architecture". Open Networking Foundation ONF SDN ARCH, Issue 1, June, 2014.
41. 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.
42. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campusnetworks. SIGCOMM Comput. Commun. Rev., 38(2):69–74, 2008.