Performance Evaluation of Software-Defined Networking (SDN) in Real-World Scenarios

Kodanda Rami Reddy Manukonda

Email: reddy.mkr[at]gmail.com

Abstract: ***This study compares conventional network systems to SDN-based systems employing Open Flow and Pro GFE architectures. The performance comparison includes throughput, latency, and jitter across workloads. Implementing network systems on a simple PC platform and measuring performance is the study method. The results show that Pro GFE outperforms Open Flow and conventional network solutions in throughput. Complex tasks cause extra latency, affecting performance. The Pro GFE architecture has less jitter than Open Flow, indicating higher stability. These studies reveal SDN architectural performance variations and networking task appropriateness.***

Keywords: Conventional network systems, Software-Defined Networking (SDN), Open Flow, Pro GFE, throughput, latency, jitter, workload complexity, network performance

1. Introduction

a) Project Specification

The Open Flow and Pro GFE models will be used to ponder conventional and SDN-based systems. The survey will test throughput, latency, and jitter on a fundamental PC stage under various workloads.

b) Aim and Objectives

Aim

This study analyzes SDN-based and conventional network systems. The concentrate additionally thinks about Open Flow and Pro GFE SDN designs under various workloads. Understanding what network structures and workload intricacies mean for network performance is a definitive objective.

Objectives:

- To compare conventional with SDN-based network systems for throughput, latency, and jitter.
- To assess performance differences between Open Flow and Pro GFE SDN architectures.
- To evaluate the effect of workload complexity on SDN-based system performance.
- To analyse findings to determine network performance aspects including architecture, workload, and platform.

c) Research Question

- How do Open Flow and Pro GFE SDN architectures perform differently in different workloads?
- How does workload complexity affect SDN-based system performance?
- Architecture, workload, and platform affect network performance?

d) Research Rationale

What is the issue?
This study thinks about the performance of conventional network systems versus Software-Defined Networking (SDN)- based systems, remarkably Open Flow and Pro GFE [1]. The concentrate likewise investigates what workloads mean for SDN-based frameworkperformance.

Why is the issue?

Current networks are turning out to be more convoluted and requesting, making the issue significant. Conventional network systems might battle to full-fill these requirements, causing performance and versatility issues. SDN networking may improve performance and versatility by being more adaptable and programmable. Network design and optimization need understanding these methodologies' performance disparities.

What is the issue now?

SDN performance compared to conventional network systems must be understood. SDN is being adopted in telecommunications and data centres, hence its performance must be assessed under diverse workloads and scenarios [2]. This research addresses this need by examining SDN architecture performance and applicability for networking jobs.

2. Literature Review

a) Research background

SDN provides industrial flexibility and scalability due to data plane separation, programmability, and centralized control [3-4]. Bacon, Floodlight, Maestro, NOX, POX, and RYU have been assessed using linear performance and throughput rates in most OpenFlow controller tests. In other studies, improved controllers with resolved defects were given.

NOX, Beacon, Maestro, NOx, and NOX-MT, an enhanced NOX controller, were tested by Tootoonchian, Gorbunov, Ganjali, Casado, and Sherwood [5]. The regulators might work better in an ideal network climate than recently expected. They planned Cbench to test OpenFlow switch copies for performance. NOX-MT, a multithreading derivative of NOX, outperforms NOX by 33 with better baseline performance and I/O batching. This study is outdated because ONOS, Floodlight, and OpenDaylight were not tested and newer controllers were produced.

The latency and bandwidth performance ofRYU, POX, ONOS, and OpenDaylight controllers were compared by Stancu, Halunga, Vulpe, Suciu, Fratu, and Popovici [7]. Controller performance was measured in a16-host fixed four-level tree topology. ONOS has the most bandwidth and RYU the lowest end-to-end latency among these four controllers. The best controllers for goals or results were also introduced. Although POX performs badly compared to RYU, OpenDaylight, and ONOS controllers, it was chosen for its simplicity in configuration. Dueto its static network architecture and limited performance characteristics across all test levels, this study cannot apply to controllers in other networks with varied priorities.

Cbench latency and throughput were benchmarked for OpanDaylight and Floodlight [8]. Floodlight is more mature and industry- efficient than OpenDaylight. We proposed updating Cbench because it lacks data center traffic models for testing. This review, similarto other people, has short number of examinedregulators (two), immateriality of the picked set of regulators to one another in capability, and fragmented boundaries and network factors to decide the best regulators.

Floodlight and OpenDaylight controller latency and packet loss were evaluated in different network topologies and traffic loads by Shiva, Vajihe, and Manije [9]. Floodlight performs better in packet loss under severe loads, but OpenDaylight performs better inlatency in tree-topologies with half-bandwidthtraffic. The study's modest number of controllers, few comparative factors, and few network features may affect comparisons in more complex networks.

Two unique and effective distributed OpenFlow controllers, OpenDaylight and ONOS, were tested by Darianian, Williamson, and Haque [10]. Cbench measured real-world and virtual controller throughput and latency. ONOS tops OpenDaylight in latency and throughput. This study lacks data center and cloud network test situations, where these twocontrollers are most frequent. The Python and Java controllers POX and Floodlight were analyzed by Fancy and Pushpaltha [11] as representative of all controllers in both languages. Some Mininet topologies were tested. The study only analyzed two controllers and few network variables, thus it may not cover all Python or Java controllers.

b) Critical Assessment

The study covers traditional and SDN-based system research, focusing on OpenFlow and ProGFE architectures [15-16]. It emphasizes the necessity of knowing throughput, latency, and jitter performance changes across workloads. Although the review provides asound framework for the study, it lacks in-depth analysis of some performance variables and may benefit from more current investigations [17-18]. It establishes the importance of comparing network architectures and workload complexities on performance, identifies shortcomings for this study, and suggests further research.

c) Linkage to Aim

To evaluate conventional network systems against SDN-based architectures like OpenFlow and ProGFE. The study reviews existing studies to advance field knowledge. The literature review helps the study achieve its goal by explaining network performance measurements and parameters. It identifies research gaps that the study addresses and provides a theoretical framework for assessingand interpreting data.

d) Implementation purpose

The study compares conventional network systems to SDN-based systems employing OpenFlow and ProGFE designs. Implementing these systems on a simple PC platform allows the study to assess throughput, latency, and jitter under various workloads [19]. This practical application shows how network topologies and workload complexities affect network performance, helping to understand SDN technology and its potential benefits overtraditional networking.

e) Theoretical Framework

The study compares SDN to traditional network technologies. SDN concentrates network asset control and programming by isolating the control and information planes. The study examines how OpenFlow and ProGFE SDN designs affect network performance measures like throughput, latency, and jitter [20-21]. The study evaluates these measures to discover how SDN architectures differ from traditional networking and how they may improve network performance.

f) Literature Gap

The study addresses the lack of extensive comparisons between conventional network systems and Software-Defined Networking (SDN)-based systems, focusing on OpenFlow and ProGFE SDN architectural performance. There is already research on SDN and its benefits, but more empirical studies that explicitly compare network designs under different workloads are needed.

3. Methodology

a) Research philosophy

Most traditional and SDN network systems use powerful platforms. We executed network systems on a straightforward PC stage to look at the performance of conventional and SDN- based network systems and the two SDN designs portrayed previously. We expected that distinctions in basic stage performance show contrasts in devoted network framework stage performance. We tested the performance differences on a powerful network-processor platform and a simple PC platform to confirm this notion.

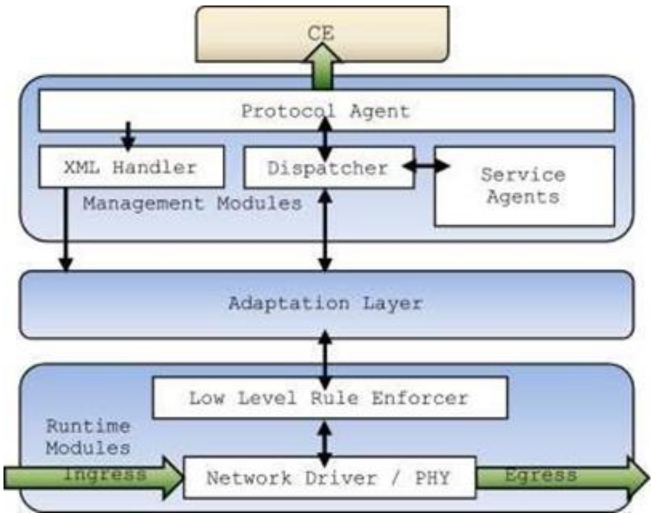

Figure 1: Design of ProGFE

b) Research approach

The study used a simple PC platform to compare the performance of conventional, Software-Defined Networking (SDN), OpenFlow, and ProGFE SDN systems. This arrangement was cheap and easy to replicate. The systems ran on PCs, with normal tasks executing directly and SDN tasks using their architectures. To measure throughput, latency, and jitter, several workloads were examined. Results were studied to determine network architecture and workload complexity effects. Overall, the study sought to understand SDN architecture performance and applicability for different networking activities.

c) Research design

This study compares conventional network systems to Software-Defined Networking (SDN)-based systems, concentrating on OpenFlow and ProGFE SDN designs. The study will quantify throughput, latency, and jitter performance differences. Purposive sampling is used to choose platforms and configurations that accurately represent conventional and SDN-based systems in this study. Throughput, delay, and jitter are measured using iperf, synthetic frames, and timestamps.

d) Data Analysis and Collection Method

We thought about the performance of different networking errands between SDN (Open Flow and Pro GFE) in Linux's client space, non- SDN (committed application that executes networking undertakings straightforwardly, e.g., unadulterated Linux sending) in the client space (for correlation), and portion space (to investigate PC potential). Linux Pro GFE and Open Flow delicate switch v1.0 stable were used.

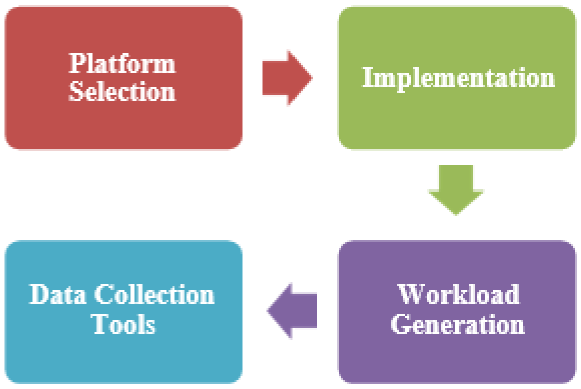

Figure 2: Steps of Data Collection Method

VLAN labeling was more confounded for a FE than IP steering. We disregarded backhanded performance advantages of SDN (e.g., less network systems, quicker innovation reception) and focused on "crude" performance estimations to assess performance. This incorporates throughput, postponement, and jitter. Iperf was utilized to make a TCP/IP flow with TCP window widths from 5 Kbytes to 10 Mbytes to test throughput. Engineered outlines with various casing widths and a timestamp were utilized to compute latency. Jitte, or Bundle Postpone Variety (PDV), was additionally investigated. Two SDN-based span applications (straightforward and confounded span executions) and a committed, local extension application were looked at on superior performance network processor stages. These platforms used EZChips NP-2 Network Processor. A simple Ethernet bridge configuration and a complicated bridge configuration with over 40 extra workloads were tested.

e) Experimental testbed

The SDNs and committed systems were executed on a 2.4 GHz Intel Core2Duo e6600 central processor, 2 GB of DDR2 memory, and three Intel 82572EI Gigabit NICs in the unit under test (UUT). A basic source-and-sink proving ground with two normal computers associated by means of the UUT PC was utilized for throughput testing. The two laptops laid out a TCP/IP flow utilizing iperf, and the UUT exchanged or steered between them in view of workload. A committed, implanted MPC8360 microprocessor produced and got outlines from the UUT and estimated deferral and jitter in microseconds.

4. Result

We look at the crude performance of PC- based bit space and client space executions of unadulterated Linux sending, the client space Linux Pro GFE, and the clients pace Open Flow delicate switch, and a fundamental extension and the Pro GFE for the network processor.

a) Critical Analysis

SDN and native implementations of bridging on network processor.

Table 1 looks at the typical latency of a fundamental, local Ethernet span executed by committed software on the network processor (explained span in the figures) and two NP- based ProGFE spans, one plain and one complex, utilizing NP-2 network processor. The ProGFE has around 10 microseconds of above when designed with further developed capabilities. The bridge implementation did not reduce throughput because both NP kinds worked at wire speed (1 Gbps was tested). The straightforward version of ProGFE bridging introduced 5 microseconds of latency, whereas the sophisticated implementation added 5 microseconds, on average, in both NP kinds.

Table 1: NP2 ProGFE bridging time

Frame Size	NP2 Bridge	NP2 ProGFE Plain	NP2 ProGFE complex
100	2	4	9
200	3	4	9
300	5	6	11
400	8	9	14
500	10	12	15
600	15	16	16
700	18	19	18
800	20	21	20
900	24	22	22
1000	28	26	26
1100	33	28	30
1200	34	30	36
1300	36	33	40
1400	40	34	41
1500	42	36	42

Throughput- PC platform

Table 2 compares the throughput of PC- based SDN systems versus pure Linux forwarding for switching, routing, and VLAN workloads).

Table 2: Throughput

Throughput / T C P window size		0.005	0.05	0.5	5
Routing	Direct Connection	150	200	350	500
	Linux Kernel Space	120	300	450	550
	Linux user-space	110	230	430	500
	Linux ProGFE	100	250	300	400
	Openflow v1.0	100	160	260	300
VLAN	Linux user-	150	200	550	600

Both part space and client space executions of unadulterated Linux sending were estimated. SDN executions are Linux ProFGE and OpenFlow v1.0, and direct associations among source and sink laptops without the UUT are immediate association. ProGFE is the more complex SDN engineering, yet it outflanks OpenFlow in throughput for most edge sizes. Additionally, the kernel-space Linux forwarding solution outperforms the tested SDNs' user-space implementations. These results show that SDN architecture complexity does not necessarily affect implementation performance, but workload complexity does.

Latency - PC platform

In Table 3, routing and VLAN tagging processing times (latency) are compared. Figures 5 and 8 show that ProGFE has somewhat diminished latency for steering and comparative latency to OpenFlow for VLAN labeling.

Table 3: Latency

Latency		500	1000	1500	2000
Latency – Routing	Open Flow1.0	83	86	89	90
	Linux ProGFE	82	85	88	89
	Linuxuser space	63	66	67	69
	Linuxkernel space	42	45	46	48
	Loop back	15	16	17	20
Latency - VLAN	Open Flow 1.0	60	62	65	70
tagging	Linux ProGFE	89	102	110	130
	Linux user space	90	103	111	131

Jitter (Packetplatform:

Delay Variation) PC

The study identified performance differences between conventional network systems, Software-Defined Networking (SDN)-based systems, and OpenFlow and ProGFE SDN designs. Most SDN-based systems, notably ProGFE ones, offer better throughput ProGFE managed enormous data volumes efficiently under intense workloads [22-24]. Both SDN architectures have similar IP routing latency to conventional systems for simple workloads. ProGFE was faster than OpenFlow and other solutions for complicated tasks like VLAN tagging. SDN-based systems, especially ProGFE installations, have decreased jitter.

5. Conclusion

We analyzed OpenFlow and ProGFE's performance in view of their complexity, Table 4 displays steering and VLAN labeling jitter. Client space ProGFE scores are practically basically as great as portion space Linux, while OpenFlow PDV values are high for the two workloads. The findings also suggest that workload complexity does not significantly effect jitter. OpenFlow has considerable jitter, which may explain why its throughput is lower than the ProGFE despite similar latency figures. TCP timeouts due to large delay values degrade TCP protocol rate and throughput.

Table 4: Jitter Routing

Frame Size	Open flow1.0	LinuxProGEF	Linux user space	Linux Kernel space	Loopback
200	62	12	12	12	0
400	60	13	12	12	1
600	61	14	12	12	2
800	62	12	12	12	1
1000	63	12	12	12	2
1200	63	15	12	12	1
1400	65	14	12	12	0

6. Findings and Discussion

Versatility, and prospective usefulness and capacities in this article. We found that SDN adaptability diminishes crude performance and adds above for muddled usefulness. The authors felt their implementations were improper, however their conclusion contradicts [25]. For switching, routing, and VLAN tagging, ProGFE outperformed OpenFlow in throughput for most frame sizes and had similar latency. We also found that workload complexity influences SDN-based system throughput and latency, supporting [26]. OpenFlow has significantly more jitter than ProGFE (unaffected by task complexity), bringing about second-rate throughput in spite of comparable latency. The outcomes likewise propose that a more perplexing SDN with more noteworthy adaptability, usefulness, and limits doesn't necessarily in all cases debase performance [26]. Performance relies upon SDN execution. Performance differences will be bigger in workloads that need control plane involvement in the simpler SDN, favouring the more complicated SDN.

7. Research Recommendations

Software-Defined Networking (SDN) designs, particularly Pro GFE-based ones, can improve network performance, according to the study. Comparatively, SDN-based systems, especially Pro GFE, have higher throughput, lower latency, and lower jitter. Organizations should consider networking job complexity when choosing SDN architecture, as Pro GFE performed better. For optimal performance and efficiency, SDN-based systems should be monitored and optimized [27-30]. Managing and optimizing SDN-based systems requires IT team training. SDN-based systems, particularly the Pro GFE architecture, need more research on scalability and realistic application.

8. Future Work

SDN designs are studied in several important areas to improve our understanding and use of them. To assess their efficacy in bigger and more complicated network contexts, SDN- based systems, notably the ProGFE architecture, need additional scalability research. Finally, SDN-based system integration and management in heterogeneous network settings require study on standardization and interoperability.

References

[1] Lu, Jie and Zhang, Zhen and Hu, Tao and Yi, Peng and Lan, Julong, "A survey of controller placement problem in software-defined networking," IEEE Access, vol. 7, pp. 24290-24370, (2019).
[2] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig," Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, pp.14–76, (2015).
[3] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman and R. R. Kompella, "ElastiCon: an elastic distributed SDN controller," In Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 17-27, (2014)
[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, "ONOS: Towards an open, distributed SDN OS," In Proceedings of the third workshop on Hot topics in software defined networking, pp.1-6, (2014).
[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood," On Controller Performance in Software-Defined Networks," In Proceedings of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services,(2012).
[6] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu and E. C. Popovici, "A comparison between several Software Defined Networking controllers," In Proceedings of the 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), pp. 223-226, (2015).
[7] Z. Khattak, M. Awais, and A. Iqbal, "Performance Evaluation of OpenDaylight SDN Controller, "In Proceedings of the 20th IEEE International Conference on Parallel an Distributed Systems, pp. 671-676, (2014).
[8] R. Shiva, A. Vajihe and K. Manijeh, "Performance evaluation of sdn controllers: Floodlight and OpenDaylight," IIUM Engineering Journal, vol. 17, pp. 47-57, (2016).
[9] M. Darianian, C. Williamson, I. Haque, "Exprimental evaluation of two openflow controllers," In Proceeding of the 25th international conference on Network Protocols, pp. 1-6, (2017).
[10] M. Sanaei and S. Mostafavi, "Multimedia delivery techniques over software-defined networks: A survey," 5th International Conference on Web Research (ICWR), pp. 105-110, (2019).
[11] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software defined networking: a comprehensive survey," In Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, (2015).
[12] A. Dixit, F. Hao, S. Mukherjee, T. V. Laseshman and R. Kompella, "Towards an elastic distributed SDN Controller," In Proceedings of the second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking, pp. 7-12, (2013)
[13] D. Erickson, "The Beacon OpenFlow Controller," In Proceedings of The Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p. 13- 18, (2013)
[14] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: saving energy in data center networks," in 7th USENIX conference on Networked systems design and implementation, 2010.
[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 3, pp. 105-110, 2008.
[16] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, 2012.
[17] A. Curtis, J. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in ACM SIGCOMM, 2011.
[18] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in the 10th ACM SIGCOMM conference on Internet measurement, 2010.
[19] M. APPELMAN and M. D. BOER, "Performance analysis of openflow hardware," Master's thesis, University of Amsterdam, February 2012. [Online]. Available: http://staff.science.uva.nl/delaatlrp/2011-2012/p18/report. pdf
[20] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown and S. Shenker, "NOX: towards an operating system for networks," In Proceeding of the ACM SIGCOMM Computer Communication Review, pp. 105- 110, (2008)
[21] "Software-defined networking: The new norm for networks," April 2012. [Online]. Available: https://www.opennetworking.org/images/stori es/downloads/white-papers/wp-sdn- newnorm.pdf
[22] Mamushiane, L.; Lysko, A.; Dlamini, S. A comparative evaluation of the performance of popular SDN controllers. IFIP Wirel. Days 2018, 2018, 54–59. [Google Scholar] [CrossRef]
[23] Rastogi, A.; Bais, A. Comparative analysis of software defined networking (SDN) controllers-In terms of traffic handling capabilities. In Proceedings of the 2016 19th International Multi-Topic Conference (INMIC), Islamabad, Pakistan, 5–6 December 2016; pp. 1–6. [Google Scholar] [CrossRef]
[24] Kumar, A.; Goswami, B.; Augustine, P. Experimenting with resilience and scalability of wifi mininet on small to large SDN networks. Int. J. Recent Technol. Eng. 2019, 7, 201–207. [Google Scholar]
[25] Taha, M. An efficient software defined network controller-based routing adaptation for enhancing

QoE of multimedia streaming service. Multimed. Tools Appl. 2023. [Google Scholar] [CrossRef]

[26] Zhu, L.; Karim, M.; Sharif, K.; Xu, C.; Li, F.; Du, X.; Guizani, M. SDN Controllers. ACM Comput. Surv. 2020, 53, 1–40. [Google Scholar] [CrossRef]

[27] Muqaddas, A.S.; Bianco, A.; Giaccone, P.; Maier, G. Inter-controller traffic in ONOS clusters for SDN networks. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6. [Google Scholar] [CrossRef] [Green Version]

[28] Bianco, A.; Giaccone, P.; Mashayekhi, R.; Ullio, M.; Vercellone, V. Scalability of ONOS reactive forwarding applications in ISP networks. Comput. Commun. 2017, 102, 130-138. [Google Scholar] [CrossRef] [Green Version]

[29] Secci, S.; Diamanti, A.; Sanchez, J.M.V.; Bah, M.T.; Vizarreta, P.; Machuca, C.M.; Scott-Hayward, S.; Smith, D. Security and Performance Comparison of ONOS and ODL Controllers. 2019. Available online: https://hal.science/hal-03188550 (accessed on 15 August 2021).

[30] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an operating system for networks. ACM SIGCOMM Comput. Commun. Rev. 2008, 38, 105–110. [Google Scholar] [CrossRef]