Article

Not peer-reviewed version

Open Source SDN Controllers – Operational and Security Issues

Aleksandra Mardaus , Edyta Biernacka , Robert Wójcik , Jerzy Domżał *

Posted Date: 30 April 2024

doi: 10.20944/preprints202404.1984.v1

Keywords: SDN; controller; open-source; network; performance; resilience; security

Article

Open Source SDN Controllers – Operational and Security Issues

Aleksandra Mardaus [1], Edyta Biernacka [1], Robert Wójcik [1] and Jerzy Domżał [1]

[1] AGH University of Krakow
* Correspondence: jerzy.domzal@agh.edu.pl
‡ The authors contributed equally to this work.

Abstract: The Software-Defined Networking concept plays an important role in network management. The central controller, which is the main element of SDN, allows to provide traffic engineering and security solutions in single and multiple-layer networks based on optical transmission. In this work, we compare selected open-source implementations of SDN controllers. Throughput and latency measurements were analyzed using the CBench program. The simulation of a link failure and a controller failure were conducted using the API provided by the Mininet network simulator. For detecting security vulnerabilities, a dedicated program, called sdnpwn, was used. The work provides an overview of the selected controllers, indicating their strengths and weaknesses. Moreover, some implemention suggestions and recommendations are presented.

Keywords: SDN; controller; open-source; network; performance; resilience; security

1. Introduction

Software-Defined Networking (SDN) is a popular approach to develop and manage of telecommunication networks. It allows network administrators to manage traffic flows and policies of the entire network from a single centralized device. The concept is implemented by separating the control plane from the data plane. The control plane determines where to send traffic, while the data plane actually forwards the traffic in the hardware. This could be a cable switch or an optical device [1].

Using a centralized entity to control the entire network eliminates the need to manually configure devices and allows making changes more efficiently. It also provides independence from equipment manufacturers, allowing the usage of devices delivered by different vendors in the network.

The SDN controller is responsible for the all intelligence and decision-making within the network. There is a wide range of solutions available, both proprietary and non-proprietary. As the main purpose of SDN is being available to everyone, the article focuses on those provisioned as open source. Their source code is publicly available, anyone can use, modify and distribute it. The three open-source controllers that were selected for comparison are: OpenDaylight [7], Floodlight [8] and Open Network Operating System [9].

In order to compare the controllers, numerous experiments have been carried out. The analysis focused on their performance, failure resistance and selected security aspects. The research allowed to observe the advantages and limitations of each controller. The obtained results allow to choose the best solution according to the needs of specific application.

2. Related Work

Three open-source controllers were analyzed in [2]. The FloodLight, OpenDayLight and Ryu controllers were evaluated. Latency, throughput, and scalability were observed using the CBench tool and the Mininet environment. The presented analysis is simple and provides only a few results. The worst results are observed for the Ryu controller. It is not possible to conclude which controller performs best based on the provided results. In our analysis, we provide more advanced results including the analysis of performance, failure resistance and selected security aspects.

A comparative study of selected SDN controllers, such as ONOS and Libfluid-based controllers (raw, base), is presented in [3]. While the authors performed the analyses for many controllers, such

 doi:10.20944/preprints202404.1984.v1

as ONOS, NOX, Floodlight, Ryu, POX ant others, the obtained results are quite simple. They (only throughput and latency) are presented in figures without confidence intervals. The lack of the statistical analysis results, in fact, that it is impossible to conclude which controller performs best. In our analysis, we provide more tests and present more, statistically credible, results.

The more advanced and complex analysis of SDNs and selected controllers is presented in [4]. The authors describe the basic functions of the main elements of the SDN architecture. Moreover, some analytical data is presented to confirm that the SDN concept continues to gain popularity among market leaders. Many controllers, such as, e.g., POX, NOX, OpenDayLight, Beacon, RUNOS, FloodLight and others have been described and theoretically analyzed. Next, a few controllers were tested in selected network topologies. For each new topology, the new mininet topology class was defined in the python file. Seven use cases were analyzed by the authors:

- Legacy Network Interoperability,
- Network Monitoring,
- Load Balancing,
- Traffic Engineering,
- Dynamic Network Taps,
- Multi-layer Network Optimization,
- Network Virtualization.

The obtained results are described by the authors. They are presented neither in figures nor as strict data in tables. However, it is possible to conclude that the OpenDayLight has shown to be the best choice in most analyzed aspects. It is recommended by the authors as the best full-detailed SDN controller. The analyses presented in our paper are, however, more advanced and possible to be verified in terms of numbers. We present more results in more aspects.

Another analysis of SDN controllers is presented in [5]. Four SDN controllers, namely: POX, Ryu, ONOS and OpenDayLight were analyzed in the Mininet simulation environment. The authors present the concept of SDN and the basic information about the controller functionality. Next, the testing methodology is described. Four layers of switches were aligned in a tree topology. 16 hosts were created. Two of the hosts were connected to a particular switch. All analyzed types of the SDN controller were observed in this topology. Two performance tests were conducted:

- a ping between hosts h1 and h16 — ten ICMP packets were sent to determine the average RTT of the packets,
- network performance between two nodes (h1 and h16) measured by generating TCP traffic with iperf and observing the througput.

The best results were observed for ONOS and OpenDayLight controllers for RTT and throughput in the switch mode tests. Our analyses are more advanced and are extended by the latency and security tests.

The authors of [6] decided to analyze various python-based SDN controllers. POX, Ryu and Pyretic were tested using the Mininet emulator. First, the authors present the main idea of SDNs and, next, the basic information of the selected controllers. Throughput and latency were measured. The experiments were conducted using VMPlayer in the Mininet environment, which was used to create a topology containing three hosts, one switch, and one controller. The ping service was used for the RTT analysis and the iperf was used to measure throughput. The results indicate that the best parameters (latency and throughput) were observed for the Ryu controller. The presented analysis was very simple. In this paper, we provide more complex simulations and present more advanced results.

All papers presented in this section are based on the SDN controllers analysis. Different types of controllers were tested, and different results were observed. Our analysis is definitely more complex. Entirely new are the security and failure resistance analyses presented in 4.5-4.7 sections.

 doi:10.20944/preprints202404.1984.v1

3. Testing Environment

The research was carried out in a virtual environment. Each controller was installed on a separate virtual machine (VM). Another VM was created to run mininet — the network simulator to be managed by the remote controller. Four virtual machines with Ubuntu 22.04 LTS Jammy Jellyfish operating system were created using VirtualBox hypervisor. Additionally, one virtual machine with Ubuntu 14.04 was created specifically for running CBench, a performance testing program, which requires the use of libraries that are not available on newer operating systems. Furthermore, a machine with Kali Linux was created to enhance the credibility of security testing.

To measure controllers' performance, Controller Benchmarker (CBench) [10] was used. It is a dedicated tool for rating the performance of controllers, which use the OpenFlow protocol [11]. CBench works by emulating a large number of switches that connect to a controller, generating PacketIn messages for each new flow. Two metrics used by CBench are latency and throughput. Throughput is defined as the rate at which the controller processes flows. It is the number of received PacketOut messages, that can be compared with the number of PacketIn messages sent by switches in a unit of time [14]. Controller latency is the time the controller needs to process a single packet. CBench measures latency by sending a single PacketIn packet to the controller and waiting for a response before sending another packet. This operation is repeated several times over a set time interval, and the average time is calculated [15].

4. Results

In this section, we present the results of carefully selected simulation experiments.

4.1. Throughput in Relation to the Number of Switches

In the first experiment, CBench was run in the Throughput Mode. The number of switches in the topology was changed in each run to values: 1, 2, 4, 8, 16. The experiment was repeated 20 times. The results are presented in Tables 1 and 2. The error bars are calculated using the t-Student distribution (confidence interval equal to 0.95). The results were visualized in Figure 1.

Table 1. Number of flows processed by Floodlight controller in 1ms.

Number of switches	1	2	4	8	16
Average	596034	496936	282894	12959	62840
Lower limit of the confidence interval	585726	452908	269954	9059	53870
Upper limit of the confidence interval	606341	540965	295834	16859	71810

Table 2. Number of flows processed by ONOS controller in 1ms.

Number of switches	1	2	4	8	16
Average	1240979	970756	500394	237401	80303
Lower limit of the confidence interval	1063536	884232	451543	218341	67789
Upper limit of the confidence interval	1418422	1057280	549245	256460	92817

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2024 doi:10.20944/preprints202404.1984.v1

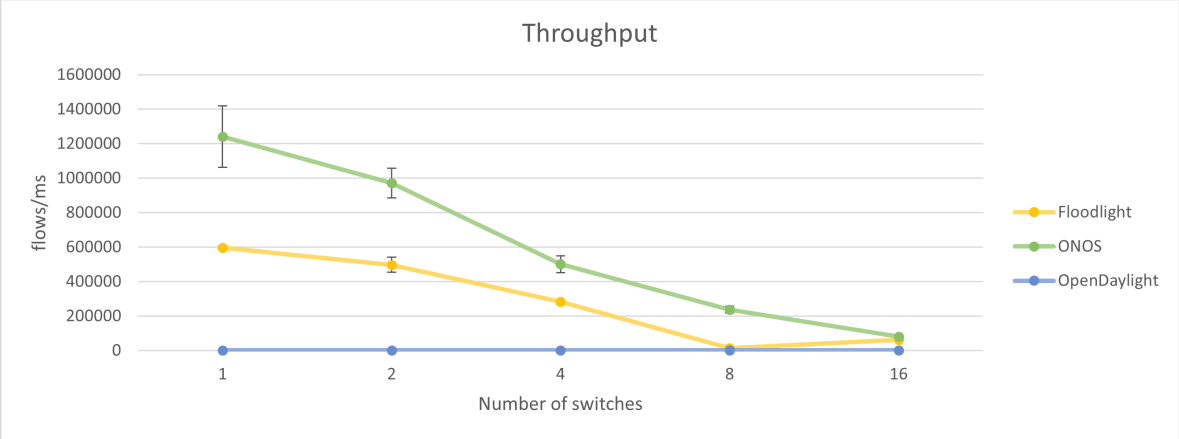


Figure 1. Throughput vs number of switches.

The throughput provided by ONOS is twice as large as for Floodlight. It was observed that for both controllers throughput decreases as the number of switches doubles. The relation between those values is proportional, resembling a linear function. It was expected and confirmed by the simulation results that the operational possibility observed for the controller decreases with the increasing number of devices it has to communicate with.

During the experiment execution, the OpenDaylight failed to provide any results. It returned zeros instead of meaningful values. However, the investigation indicated that this problem has already been observed by the authors of the article [**?**] and is known as the "all-zeros problem". Since the CBench program is unable to measure the throughput of the OpenDaylight controller, it was not included in the calculations.

4.2. Throughput vs. Number of Unique MAC Addresses on a Switch

The second experiment involved measuring throughput in relation to the number of unique MAC addresses in the network. CBench was again run in the Throughput Mode, repeated 20 times, and the number of MAC addresses was altered within a range from 100 to 1 000 000. Tables 3 and 4 and Figure 2 show the results.

Table 3. Number of flows processed by Floodlight controller in 1ms.

MAC addresses	100	1 000	10 000	100 000	1 000 000	10 000 000
Average	822307	791187	781153	743029	92608	29153
Lower limit of the conf. int.	770540	766370	767106	728531	64880	19111
Upper limit of the conf. int.	874075	816005	795199	757528	120336	39195

Table 4. Number of flows processed by ONOS controller in 1ms.

MAC addresses	100	1 000	10 000	100 000	1 000 000	10 000 000
Average	1057122	1007840	1048151	1166272	343138	135972
Lower limit of the conf. int.	829629	761532	951346	967048	79898	107705
Upper limit of the conf. int.	1284614	1254148	1144955	1365495	606377	379649

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2024 doi:10.20944/preprints202404.1984.v1

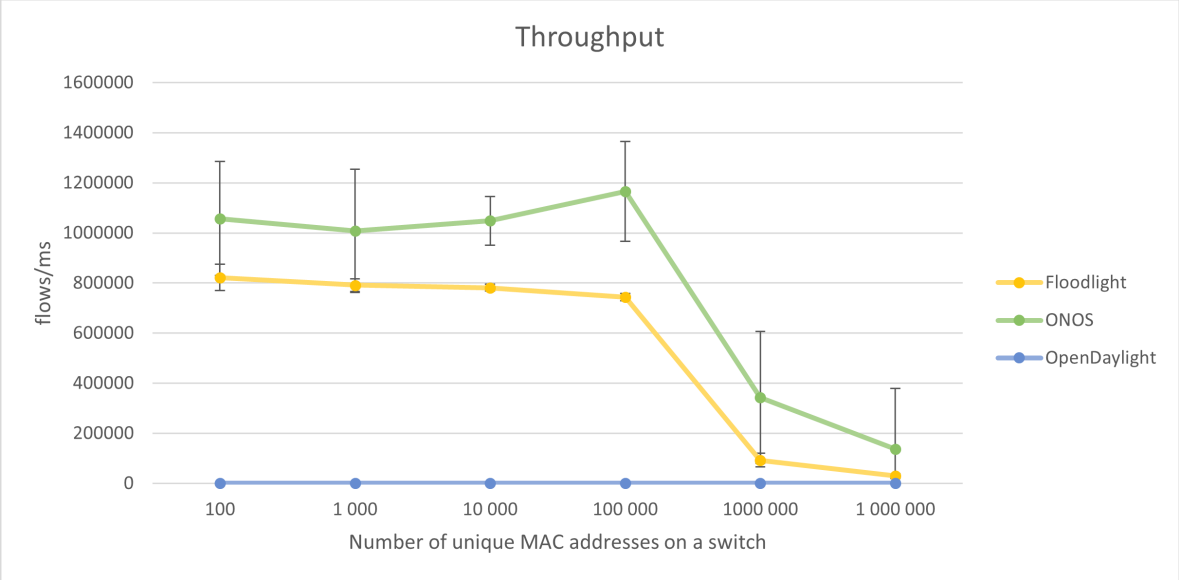


Figure 2. Throughput vs number of unique MAC addresses on a switch.

Figure 2 shows the changes in throughput in relation to the number of unique MAC addresses on the switch. OpenDaylight again failed to return valid results. ONOS was observed to provide higher throughput than Floodlight. Throughput did not change significantly when the number of unique MAC addresses in the switch was between 100 and 100 000. A significant decline in throughput occurred when the number of unique MAC addresses on the switch exceeded 100 000. However, this may not be the result of the low performance of the controller but caused by the limitations of the simulation environment. The measurements associated with the ONOS controller have a larger standard deviation and, consequently, the confidence intervals are much larger than for Floodlight.

4.3. Latency vs Number of Switches

In this section, the CBench program was run in the Latency Mode. The first experiment involved measurements of delay in relation to the varying number of switches. The results are shown in Tables 5, 6 and on Figure 3.

Table 5. Latency in milliseconds for OpenDaylight controller.

Number of switches	1	2	4	8	16
Average	1.222	2.133	3.471	4.149	4.915
The lower limit of the confidence interval	1.196	2.084	3.381	3.890	4.425
The upper limit of the confidence interval	1.249	2.182	3.562	4.408	5.404

Table 6. Latency in milliseconds for Floodlight controller.

Number of switches	1	2	4	8	16
Average	0.862	1.613	1.971	2.686	3.467
The lower limit of the confidence interval	0.703	1.544	1.573	2.492	3.155
The upper limit of the confidence interval	1.021	1.681	2.370	2.880	3.778

Table 7. Latency in milliseconds for ONOS controller.

Number of switches	1	2	4	8	16
Average	0.595	0.633	0.671	0.75	0.857
The lower limit of the confidence interval	0.578	0.614	0.660	0.720	0.651
The upper limit of the confidence interval	0.611	0.653	0.682	0.779	1.063

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2024 doi:10.20944/preprints202404.1984.v1

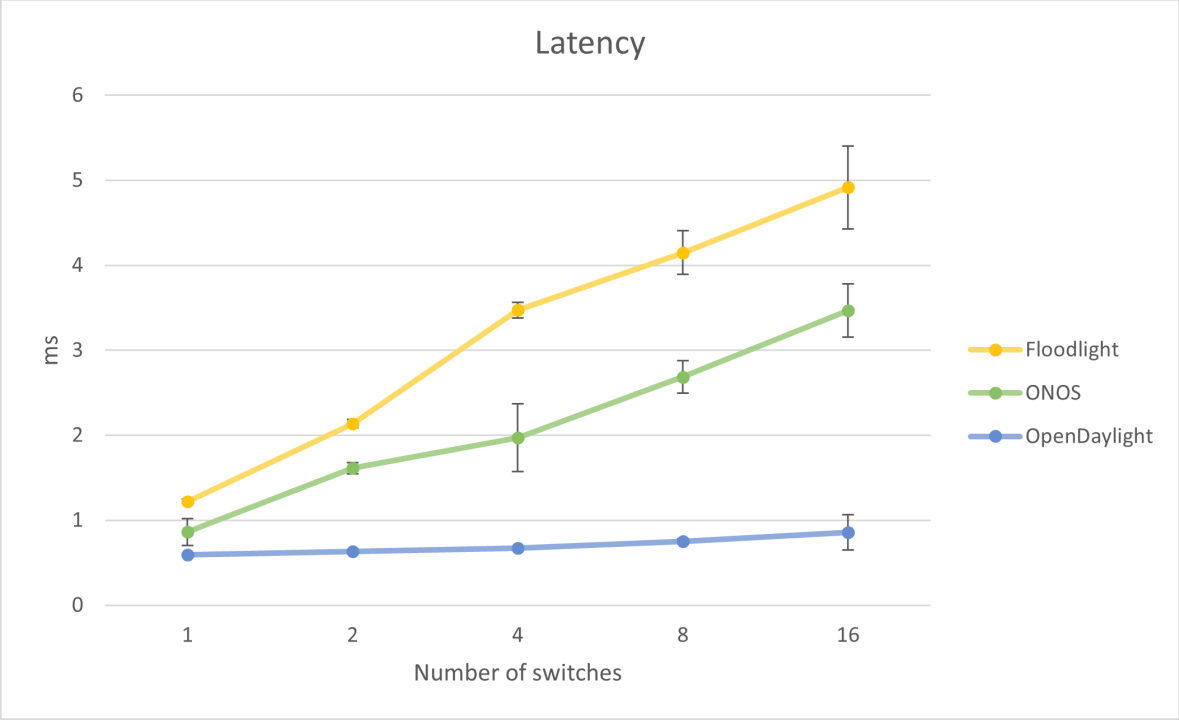


Figure 3. Latency vs number of switches.

Figure 3 shows the relation between the latency and the number of switches in range from 1 to 16. OpenDaylight provides the lowest latency. Its values are approximately constant, which leads to the conclusion that they do not depend on the number of switches. The shape of the graphs for Floodlight and ONOS switches is close to a linear function when increasing with the number of switches. Floodlight achieved the highest latency values of those three compared controllers.

4.4. Latency in Relation to the Number of Unique MAC Addresses on a Switch

The last measurement using CBench conveyed the measurement of latency against the number of unique MAC addresses in the network. The results are presented in Tables 8, 9, 10, and in Figure 4.

Table 8. Latency in milliseconds for OpenDaylight controller.

MAC addresses	100	1 000	10 000	100 000	1000 000	1 000 000
Average	1.206	1.239	1.321	1.268	1.146	1.298
The lower limit of the conf. int.	1.192	1.196	1.285	1.236	1.077	1.263
The upper limit of the conf. int.	1.219	1.283	1.357	1.300	1.215	1.332

Table 9. Latency in milliseconds for OpenDaylight controller.

MAC addresses	100	1 000	10 000	100 000	1000 000	1 000 000
Average	1.022	0.928	0.942	0.923	0.964	0.874
The lower limit of the conf. int.	0.968	0.786	0.838	0.820	0.852	0.728
The upper limit of the conf. int.	1.075	1.070	1.047	1.026	1.075	1.020

Table 10. Latency in milliseconds for OpenDaylight controller.

MAC addresses	100	1 000	10 000	100 000	1000 000	1 000 000
Average	0.599	0.6	0.603	0.586	0.606	0.557
The lower limit of the conf. int.	0.591	0.588	0.591	0.570	0.599	0.534
The upper limit of the conf. int.	0.608	0.612	0.615	0.602	0.613	0.579

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2024 doi:10.20944/preprints202404.1984.v1

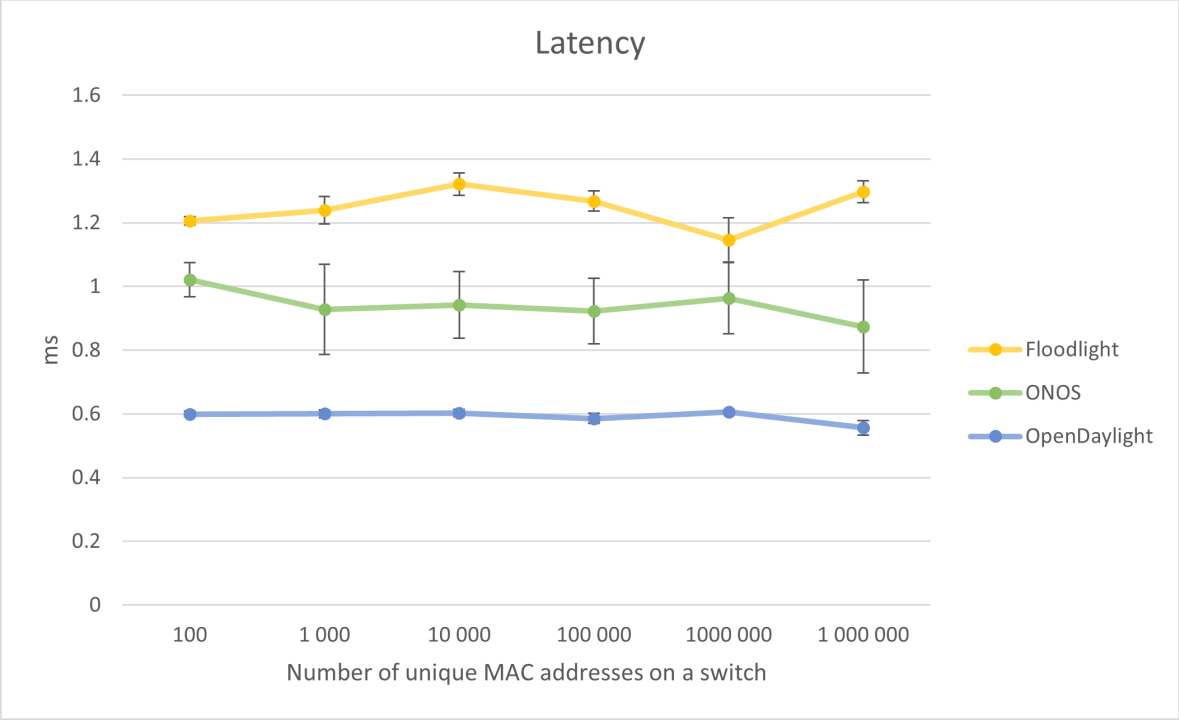


Figure 4. Latency in relation to the number of unique MAC addresses on a switch.

It can be observed in Figure 4 that there is no specific relation between latency and the number of unique MAC addresses in the switch. Considering the measurement uncertainties visible as error bars in the graph, the function describing this relation can be compared to a linear function with a constant value. The lowest latency is achieved by OpenDaylight, while the highest by Floodlight. It can be concluded that the latency does not depend on the number of unique MAC addresses on the switch.

4.5. Controller Failure

A simulation of a controller failure was conducted in Mininet. A minimal topology was built — two hosts connected to a switch, as shown in Figure 5.

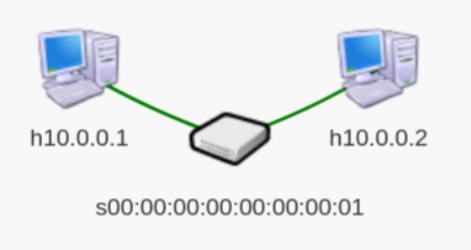


Figure 5. Topology used in simulation of controller failure.

The time of a controller failure was measured as the time observed since the loss of connectivity between h1 and h2 (caused by the controller being turned off) to the restoration of reachability. It was observed during the experiment that, when using the OpenDaylight controller, turning off the controller did not impact the network connectivity. The switches did not remove entries from their flow tables despite the loss of communication with the controller. Such behavior can be considered beneficial, as network operation is maintained despite the failure of the controller.

However, to include OpenDaylight in the comparison, flows were manually removed from the switch after each simulated failure. The results obtained are shown in Figure 6.

 doi:10.20944/preprints202404.1984.v1

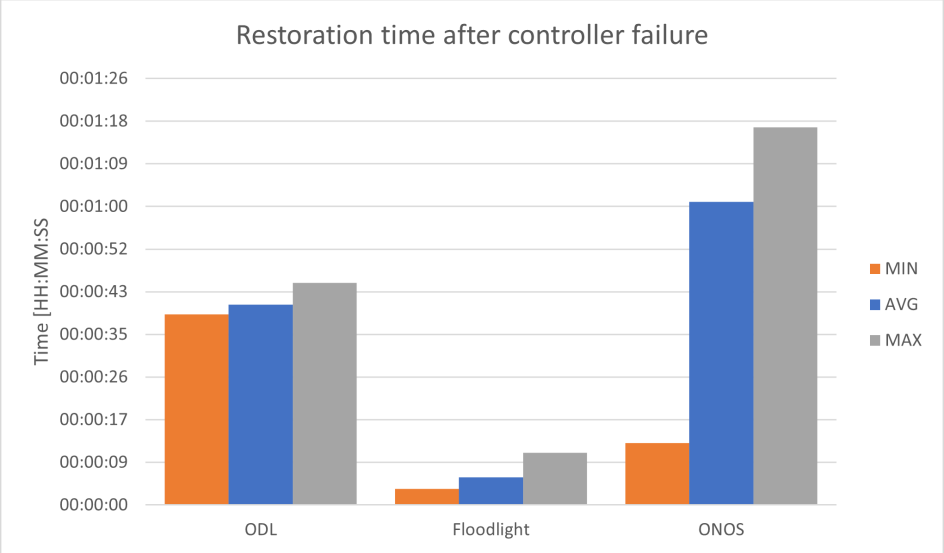


Figure 6. Network restoration time after controller failure.

The majority of restoration time is the time it takes the controller to reboot. It also includes a significantly smaller amount of time from the time instant when the controller switches back on to the time instant when connectivity between devices on the network is restored. The results shown in Figure 6 contain the sum of both time periods. ONOS needs a very long time to start up, with an average of one minute and a maximum of one minute and 16 seconds, which is due to the fact that it is the most advanced controller and needs the longest time to load its functionalities. OpenDaylight obtained better results than ONOS, achieving a minimum of 38 and a maximum of 44 seconds. Floodlight performed best, having the restoration time of average 5 seconds and a maximum of 10 seconds.

4.6. Link Failure

To measure restoration time after a link failure, a script written in Python using the Mininet API was used. A network consisting of 4 switches was constructed, with one host connected to each switch (Figure 7). Additionally, a backup link was added between the two unconnected switches.

Link failure restoration time was measured as the time from the loss of connectivity between h1 and h2 caused by disabling the direct link between them, to the restore of connectivity after switching to the backup link, as shown in Figure 8.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 April 2024 doi:10.20944/preprints202404.1984.v1

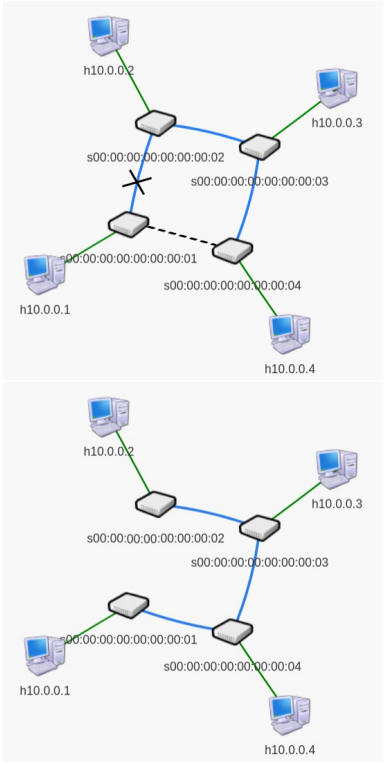


Figure 7. Link failure simulation. (**a**) Topology before link failure; (**b**) Topology after link failure.

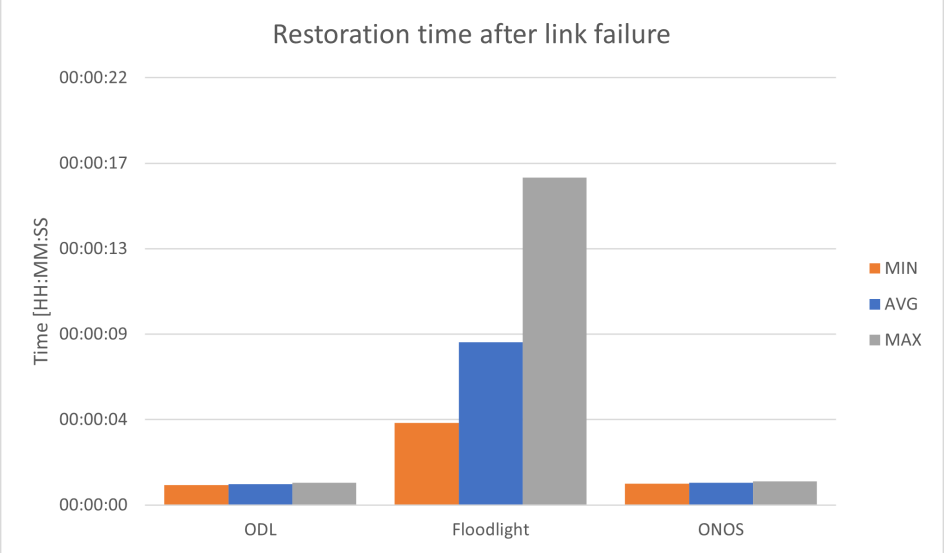


Figure 8. Network restoration time after link failure.

According to the study, in the event of a link failure, the OpenDaylight and ONOS controllers take approximately 1 second to switch to an alternative route. Floodlight, on the other hand, requires an average of 8 seconds to do so. In the event of a controller failure, Floodlight achieved the best recovery time. It restarts much faster than the other controllers — an average of 5 seconds, while OpenDaylight took approximately 40 seconds and ONOS as long as a minute.

4.7. Security

For security testing, a dedicated tool was used — sdnpwn. It is a penetration testing program that allows known attacks to be performed on selected controllers to identify their vulnerabilities.

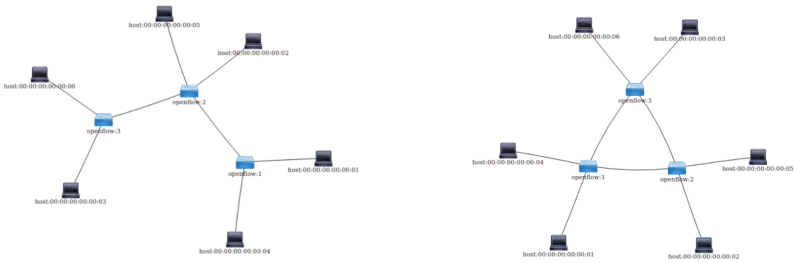
 doi:10.20944/preprints202404.1984.v1

4.7.1. Detecting the Presence of a Controller in the Network

The first experiment was conducted to determine whether it is possible to detect and identify a controller in the network. This was verified using the sdnpwn controller-detect module. The OpenDaylight controller was detected, though its version was identified incorrectly. Nitrogen version was recognized as Hydrogen — the first version of the controller. The ODL Nitrogen run was incorrectly recognized as Hydrogen. Other versions (Lithium, Boron) were also examined, providing the same result. This may indicate the inability to determine the version without having access to the controller. The port number on which the GUI is located has been correctly identified, which may potentially pose a risk. An unauthorized user, knowing on which port the GUI is available, may try to log into it. If the administrator leaves the default password or sets not complex password, an attacker will be able to perform network changes from the GUI. Both Floodlight and ONOS were recognized correctly. In both cases, the version was not specified, leading to the conclusion that sdnpwn does not recognize versions at all.

4.7.2. LLDP Replay

In SDN networks, the LLDP protocol is used to detect links between devices. The controller recognizes the topology of a given network based on the LLDP packets received from the switches. The LLDP Replay attack consists of intercepting an LLDP packet generated by the selected switch and sending it from another location in the network. This results in a message being sent to the controller containing information about a link that is not actually present in the network. The controller includes this link in the topology and attempts to send packets through it, which will then fail to reach the recipient.



(a) Before the attack. **(b)** Topology after link failure.

Figure 9. LLDP attack simulation.

Figure 9 shows the topology used when performing the LLDP Replay attack. During the experiment, it was observed that ONOS is the only one of the compared controllers with protection against the LLDP Replay attack. Floodlight and OpenDaylight do not verify LLDP packets and may accept incorrect topology information, which may result in packets being lost or sent to the wrong recipient.

4.7.3. ARP Poisoning

ARP Poisoning is a type of attack in which an attacker sends forged ARP messages to another device on the network. The device places an entry in the ARP cache that binds an IP address from an actual device on the network to the attacker's MAC address. The attacker can then receive messages directed to the other device.

This is not a type of attack specific to SDN networks, but a centrally managed network should implement mechanisms to protect against such an attack. The controller should monitor all ARP traffic on the network and prevent it from being 'poisoned'. Methods to prevent an ARP Poisoning attack are described in [13].

 doi:10.20944/preprints202404.1984.v1

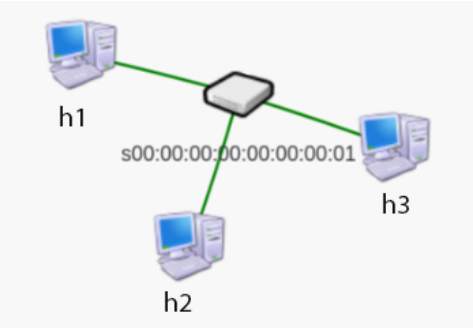


Figure 10. Topology used in ARP Poisoning attack.

The attack was carried out using sdnpwn's dp-arp-poison module. It allowed the MAC address of host h2 in the ARP table h1 to be replaced with the MAC address of the attacker (h3), causing all traffic destined for h2 to be directed to h3 instead of h2. This procedure was carried out successively for networks with the OpenDaylight, Floodlight and ONOS controller. In each case, the attack was successful. The lack of protection against ARP Poisoning means that other network attacks such as eavesdropping, Man-In-The-Middle and MAC Flooding are possible. This poses a serious threat to network security.

5. Discussion

The OpenDaylight controller development slowed down recently due to an insufficiently large community of its programmers. Its successive versions differ significantly in terms of supported modules. It performs well in latency measurements, but fails to provide any throughput measurements through the Cbench tool.

Floodlight offers acceptable performance and provides stable operation and short recovery times. It is the lightest of the compared controllers and also has the fewest known vulnerabilities described in CVE.

ONOS offers the best performance of those three controllers, providing throughput twice as big as Floodlight. However, it performs worst in terms of response time in the event of failure. It is the only controller that provides protection against LLDP Replay.

The obtained results lead to the conclusion that none of the compared controllers is objectively the best. Each one has its advantages and disadvantages. The decision on choosing the best implementation for a given network should be made individually by the network administrator, considering the observations described in this paper.

Author Contributions: "Conceptualization, A.M. and J.D.; methodology, A.M. and J.D.; software, A.M.; validation, J.D., E.B. and R.W.; formal analysis, A.M.; investigation, A.M., J.D., E.B and R.W; resources, A.M and J.D.; data curation, A.M.; writing—original draft preparation, A.M. and J.D.; writing—review and editing, E.B and R.W.; visualization, A.M.; supervision, J.D, E.B and R.W.; project administration, J.D.; funding acquisition, J.D., E.B and R.W. All authors have read and agreed to the published version of the manuscript."

Funding: This work was supported by the Polish Ministry of Science and Higher Education with the subvention funds of the Faculty of Computer Science, Electronics and Telecommunications of AGH University.

Data Availability Statement: Data supporting reported results can be found at: https://drive.google.com/drive/folders/1EoChOKS9FVVVy298_Rm3TJKCs-LjeYtT?usp=sharing

Conflicts of Interest: "The authors declare no conflict of interest."

References

1. Mirosław Kantor, Edyta Biernacka, Piotr Boryło, Jerzy Domżał, Piotr Jurkiewicz, Miłosz Stypiński, Robert Wójcik, *A survey on multi-layer IP and optical Software-Defined Networks* Computer Networks 162 (2019) 1.
2. Daniel Haro Mendoza, Luis Tello-Oquendo, Luis A. Marrone, *A comparative evaluation of the performance of open-source SDN controllers*, Latin-American Journal of Computing, issn: 1390-9266, 2020.

 doi:10.20944/preprints202404.1984.v1

3. Ola Salman, Imad Elhajj, Ayman Kayssi, Ali Chehab, *SDN Controllers: A Comparative Study*, Conference: 2016 18th Mediterranean Electrotechnical Conference (MELECON), 2016.
4. Neelam Gupta, Mashael S. Maashi, Sarvesh Tanwar, Sumit Badotra, Mohammed Aljebreen, Salil Bharany, *A Comparative Study of Software Defined Networking Controllers Using Mininet*, Electronics, 2022.
5. Alexandru L. Stancu, Simona Halunga, Alexandru Vulpe, George Suciu, Octavian Fratu, Eduard C. Popovici, *A comparison between several Software Defined Networking controllers*, International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service (TELSIKS), 2015.
6. Karamjeet Kaur, Sukhveer Kaur, Vipin Gupta, *Performance analysis of python based openflow controllers*, 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016), 2016.
7. OpenDaylight documentation *https://docs.opendaylight.org/en/stable-argon/*
8. Floodlight documentation *https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview*
9. ONOS documentation *https://opennetworking.org/onos/*
10. CBench documentation *https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343657/Cbench+New*
11. C. Laissaoui, N. Idboufker, R. Elassali, K. El Baamrani *A measurement of the response times of various OpenFlow/SDN controllers with CBench*, IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-2, DOI:10.1109/AICCSA.2015.7507203.
12. Mohamed Elmoslemany, Adly Tag Eldien, Mazen Selim, *Performance Analysis in Software Defined Network (SDN) Multi-Controllers*, Delta University Scientific Journal, 2023, volume 6, issn 2636-3046, doi 10.21608/dusj.2023.291037.
13. Zawar Shah, Steve Cosgrove *Mitigating ARP Cache Poisoning Attack in Software-Defined Networking A Survey*, Electronics, 2019.
14. Ayman Haggag *Benchmarking and Performance Analysis of Software Defined Networking Controllers in Normal and Failsafe Operations using Multiple Redundant Controllers*, Turkish Journal of Computer and Mathematics Education, 2021.
15. Zuhran Khan Khattak, Muhammad Awais, Adnan Iqbal *Performance evaluation of OpenDaylight SDN controller*, 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS),2014, 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS).