Journal of Computer Science

Original Research Paper

Efficiency Assessment of Software-Defined Networking for Real-Time Network Systems

[1]Neelam Gupta, [1]Sarvesh Tanwar and [2]Sumit Badotra

[1]Amity Institute of Information Technology, Amity University Uttar Pradesh, Noida, India
[2]School of Computer Science Engineering and Technology, Bennett University, Greater Noida, India

Article history
Received: 14-02-2024
Revised: 01-03-2024
Accepted: 08-03-2024

Corresponding Author:
Sarvesh Tanwar
Amity Institute of Information Technology, Amity University Uttar Pradesh, Noida, India
Email: dr.sarveshtanwar@gmail.com

Abstract: Software Networking (SDN) is growing in popularity due to its benefits, which include portability, mobility, analytics, and ease of creation. But it needs to be adequately shielded from security risks. One of the main vulnerabilities to the SDN network is the Distributed Denial-of-Service (DDoS) attack. To fulfill the demands of the complex and demanding security concerns of today, a new network philosophy is needed. Because SDN relies on a central controller, it has a single point of attack and failure. The present research monitors and analyses network traffic coming from switches, host computers, emulators, and wireless access points using a multi-vendor packet sampling technique using sFlow. In the process, we have clarified the usefulness and efficiency of the recommended strategy, which makes use of SDN controllers for the detection and mitigation of DDoS flooding attacks. The outcomes also demonstrate that the ODL controller outperforms the other remaining controllers in terms of load-shedding efficiency and flow setup latency. According to TCP bandwidth measurements, the ODL controller performs better in terms of processing power and jitter than the remaining controllers due to its higher computational complexity. These test results indicate that the jitter performance of all controllers is comparable. Overall analysis indicates that ODL is more reliable than other controllers in our scenario.

Keywords: SDN, DDoS Attacks, sFlow, Controllers, Jitter, Latency, Throughput

Introduction

The new SDN architecture separates the core networks' control logic from the fundamental routing and switching components, which act as packet forwarders (specified by the controller). Due to its advantages in terms of capacity, versatility, surveillance, and simplicity of innovation, SDN (Ali *et al*., 2023) has become more and more popular recently. Network control logic is no longer embedded in logically centralized controllers, making it easier for organizations to deploy and manage their own networks. This separation turns network switches into straightforward packet forwarding devices, adding flexibility, speed, and programmability, in Fig. 1. Approximately 64% of people on the planet utilize the internet at least once a day this is because they are connected to the internet via their smartphones or computers. Since there have always been security threats on the internet, there has been a noticeable growth in concern over internet security over the past few years.

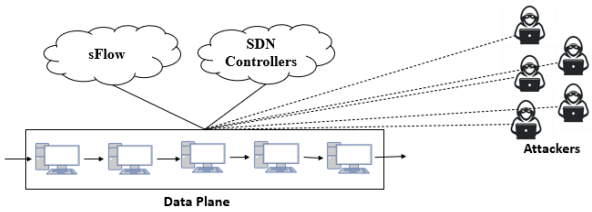

Fig. 1: SDN architecture

Trojans, worms, port scanning, and denial-of-service assaults are just some of the cyber-security concerns that have come to the researcher's attention. In a DDoS attack, the attacker (Anyanwu *et al*., 2023) probes the network for holes and inserts the Trojan horse virus into the software applications without the victims' knowledge. By copying this malicious malware onto other network-connected devices, the intrusive party builds a force of hacked computer systems that they may command to launch attacks. The increased use of communications (Rozam and Riasetiawan, 2023) networks, which enable

users to connect at any time and almost anywhere, has led to an increase in traffic demand. Unprecedented amounts of data traffic are being produced by the spread of numerous smart devices and apps as well as the development of numerous network technologies. DDoS attacks cannot be stopped by using conventional security methods such as intrusion detection systems, firewalls, and router access control lists. Due to reasons including decentralization, a lack of collaboration among ISPs, structural changes, collateral damage, outmoded tactics, and deployment challenges, there is presently no appropriate solution for identifying these attacks. Unlike earlier research that uses sniff, machine learning, etc., this research presents a sFlow tool based on information theory for distributed detection and mitigation of DDoS attack types. The goal of this strategy is to enhance these threats' detection and mitigation. A further benefit of the study is that, unlike most currently available research solutions, it has not made extensive use of real-time detection methodologies or worked with SDN controllers as extensively.

Both researchers and organizations have recently shifted their attention toward creating networks that are more reliable, scalable, and secure. In contrast to the stable and decentralized ecosystem of traditional networks (Tao *et al*., 2023), SDN developments aim to create a more dynamic and centralized nature of the network. IP-based networks make it difficult to enforce necessary regulations and reconfigure network devices. A new network design called SDN offers hope for effective network infrastructure. With centralized controllers, worldwide network monitoring, and on-demand traffic forwarding rule development, it can improve a network's security. Vertical integration is another feature of these networks' rigidity. SDN still faces issues with data security, manageability, and maintainability, among other issues. Security is the most pressing of all these issues. In this research, Gupta *et al*. (2023), add an effective and scalable mechanism to these features for executing anomaly detection and mitigation in SDN infrastructures. Anomaly detection techniques are renowned for finding both harmful and benign network patterns. Using sFlow data gathering capabilities on edge switches, we want to both enforce mitigation policies and detect network attack trends in real-time. Flow-based anomaly detection techniques have been used in numerous studies.

SDN controllers are not fully or completely compared in the numerous research publications that describe them. Many earlier studies have examined centralized OpenFlow controllers or contemplated developing a new controller. Very little research has been done on the ODL, ONOS, POX, NOX, Floodlight, and Ryu's performance in terms of burst rate, latency, throughput, Round Trip Time (RTT), jitter, and bandwidth. The authors expressly ignore the numerous other factors that would be of interest to an industrialist in favor of concentrating on the controllers' path restoration and software dependability. However, the experiment design deviates from the methods used in this publication and the research only takes a few topologies into account. All SDN controllers (Gupta *et al*., 2022a) work in ICMP DDoS attack situations (topologies in attacks using sFlow). It is crucial to assess new releases of these controllers to better comprehend the performance enhancements. We are expecting that this study will shed light on how these controllers work. All above these open-source SDN controllers are the most popular in terms of performance and acceptance. Due to the significance of the controllers in SDN, the performance of each controller is assessed in this study in terms of latency, initial and average ping delay, jitter, and throughput. sFlow is used to measure both controllers' latency, initial and average ping delays, jitter, and throughput while taking into account topologies for the network. After all the controllers (Gupta *et al*., 2022b) were evaluated, it was found that ODL provided the best responses across the board. When all of them were evaluated individually, it was discovered that the trial's findings demonstrated that ODL outperformed other remaining controllers based on certain criteria. This has the effect of making the ODL controller the best of all the controllers. This research can assist many academics and businesspeople in deciding which of the two controllers to use in various application settings.

The research is on DDoS flood attacks that target hosts with controllers and the architectural layout of a modular mechanism for SDN systems that enables anomaly detection and mitigation. We compared the outcomes of all the controllers using the sFlow tool. Additionally, we examine the performance and applicability of our suggested mechanism in relation to other well-known anomaly detection and mitigation algorithms described in the below section. Performance evaluations utilizing actual traffic traces confirm the scalability and efficiency of the suggested sFlow. It is used to measure both controllers' latency, initial and average ping delays, jitter, and throughput while taking into account topologies for the network.

Materials

The latency, ping delays, jitter, and throughput of the networks were assessed during all trials for the various controllers evaluated in two standalone test configurations, have been briefly discussed in this section. We discussed the effectiveness and efficiency of the proposed approach using ODL and Ryu controllers in detecting and mitigating of DDoS flooding attacks.

Methods

We used a 1.70 GHz Intel(R) Core (TM) i3-4005U CPU with two cores, four logical processors, and eight

gigabytes of RAM. The VirtualBox program generated various virtual machines with different names such as controllers, sFlow, Kali Linux, and Emulation. Using the Layer's switch, these devices are directly linked to the emulation device. All virtual machines (Yungaicela-Naula *et al*., 2022) were configured using the VMware Workstation VirtualBox software. The most popular testing tool for SDN controllers is Emulator, which allows for the creation of a virtual network. The emulator was utilized to establish network topologies, which were then launched by the primary SDN controller responsible for the entire network. The hardware specs of each machine used for the trials. For all controllers, we used OpenFlow protocol in various versions such as 1.3, 1.1, and 1.0, respectively for various controllers. An attack detection tool is one of the components of the Kali Linux (Mehra and Badotra, 2022) operating system for efficient DDoS operations. Twenty-five switches, thirty hosts, and one SDN controller organized with sFlow, running a Linux OS instance, make up the simulation tool. Each switch generates synthetic network traffic by randomly distributing both genuine clients and attackers. In order to detect infrastructure layer attacks, this testbed has been used to replay traffic traces, simulate user behavior quantify the impact of DDoS attacks on controllers using sFlow, and generate rule sets that are nearly identical to real ones for validation purposes.

Kali Linux is installed to generate DDoS attack traffic, as botnets are the primary method for modern DDoS attacks. The traffic generators were used to provide authentic background and typical network traffic profiles. After the topology was established, the connectivity was checked (Alhijawi *et al*., 2022) by running the ping command first. These OVS support OpenFlow, the most widely used communication protocol. All controllers underwent performance tests in the simulation to compare their results. The values are obtained by detecting and mitigating DDoS attacks on the host, followed by an analysis of the data using the sFlow tool. It serves as a user interface for changing the numerous flow table entries that these OVS possess. SDN controllers are being bombarded with data packets by the penetration tool, which is sending 8,000 extra data packets per second. It is necessary to take into account both the type of DDoS attack and the date of the controllers' failure when evaluating this metric. Further, we connected each switch module to the sFlow agent in order to compute real-time detection and effectiveness metrics using the network-based software tool, sFlow.

ICMP DDoS flood attacks are created using the Hping3 program. You can send packets that have been altered in terms of volume, amount, and segmentation to overwhelm the target and evade attacks. Using the hping3 program, you can test security or capabilities amount and segmentation to overwhelm the target and evade attacks. Using the hping3 program, you can test security or capabilities. Hping3 may be helpful for security or capability testing, as it allows you to send large numbers of packets over a secure network. We use sFlow, an SDN technology, to enhance the effectiveness of DDOS mitigation. The sFlow methodology for detecting attackers involves capturing and adding up the incremental flow from each client for analysis by the sFlow collector. Firewalls and intrusion detection systems are two techniques for protecting systems from attackers. They are not well suited to counteract DDoS attacks. When the sFlow collector (Prasad *et al*., 2022) identifies certain traffic as an attacker, the OpenFlow controller modifies the rule in the OpenFlow table. Therefore, we can immediately detect and prevent flooding attacks by combining aggregate flow using sFlow and blocking traffic using OpenFlow. To prevent attacks by restricting attack traffic, the OpenFlow controller muddles the rules a bit.

After the controllers had established communication, simulated performance tests were run on each of them to compare the results. After generating the topology by integrating the emulator with sFlow, the connection was confirmed by first executing the ping command. For controllers, it should be started first, followed by the sFlow script in a separate terminal. Following the creation of the controller-based topologies, the connectivity will then be checked, and using the pingall command, all hosts will be connected. The DDoS attack (Kumar *et al*., 2022) will then be launched by first creating normal traffic and data transmission under the topology and when it appears that the topology is functioning appropriately, use the sFlow tool to detect and mitigate the flood attack. The graph shows how data is sent while a host is under a DDoS attack from malicious packets. The data flow is evidently occurring both before and after the attack.

The sFlow-RT controller uses set flow and set thresholds to mitigate DDoS attacks. Since the objective is to filter flows, the set flow filter defines egress flows and the set threshold filter shows the specific access ports that only have thresholds applied to them.

A malicious node coupled with regular traffic arrived at the edge switch in the flowchart, in Fig. 2; the switch continuously sends information to the sFlow RT controller. An example of how a switch is protected from attacks shows that traffic on the protected hosts is blocked while normal traffic continues. An OpenFlow rule is sent to the switch to block the traffic when an attack is discovered. Attack traffic is quickly identified and eliminated with the aid of the controller using sFlow-RT (Udhaya Prasath *et al*., 2022). Normal traffic is shown to be unaffected. SDN applications for traffic engineering are developed and implemented in existing networks using the open-source sFlow-RT software platform.

Based on the popular sFlow and OpenFlow protocols, it is a fabric controller with an open, standards-based JavaScript development environment. Mitigation script logs (Batool *et al*., 2022) are once the attack has been stopped. When the attack is stopped and the mitigation block is activated, it resumes its normal course. Normal data flow resumes after that point without any issues. However, the mitigation rules are now blocking the attack host every 8 sec to thwart any potential future attacks on this architecture.

The Proposed Algorithm

The flow buffer searches for rules matching incoming packets, initiated by the flow table. If a rule is found, the packet can be sent to the OpenFlow controller or routed to an output port, with the controller overseeing the flow table. The OpenFlow protocol (Jiang *et al*., 2022) is utilized by controllers and switches to interact, offering an interface for configuration, monitoring, and data plane flow tables for packet forwarding. It provides an interface for managing packets, with received packets indicating the number of packets matching a flow. The OpenFlow controller provides network resource management data by automatically collecting metrics at predefined intervals. It allows the categorization of traffic as regular or intrusive based on the collected data. The effectiveness of mitigation depends on the length of the collection process. A long time interval may miss the initial stages of an attack, limiting mitigation time, while a short time interval increases the detection mechanism's overhead.

Selection of Parameters

Threshold: Compute the threshold $(a = 2k)$ of each flow on each switch using the formula:

$$a(s\_i,j) = -p\_i,j * log(p\_i,j \,/\, b\_i) \quad (1)$$

Average threshold: Compute the average threshold for each switch:

$$AvgThreshold(s\_i) = (1 \,/\, |S\_i|) * \sum (j = 1 \; to \; |S\_i|) \; a(s\_i,j) \quad (2)$$

The standard deviation of the threshold: Compute the standard deviation of the threshold for each switch:

$$StdDevThreshold(s\_i) = sqrt((1 \,/\, |S\_i|) * \sum(j = 1 \; to \; |S\_i|) \; (a(s\_i,j) - AvgThreshold(s\_i))\text{^}2) \quad (3)$$

Total threshold: Identify a threshold value:

$$Total\ Threshold = AvgThreshold(s) + k * StdDevThreshold(s) \quad (4)$$

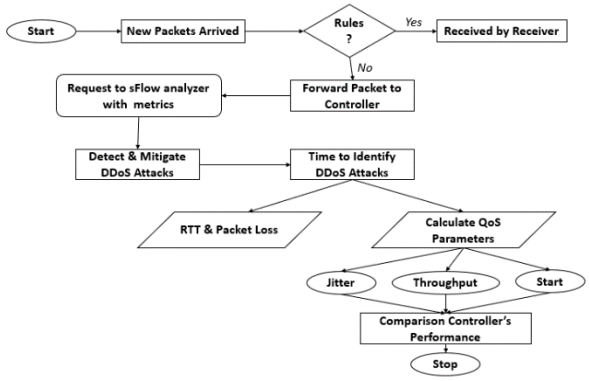

Fig. 2: Flowchart of SDN controller performance modeling and analysis

The constant value *k* is chosen based on the expected sensitivity of the DDoS detection system and network setup. Switches are continuously monitored for their threshold levels (Mishra and Gupta, 2022) and if their value exceeds the threshold, they are considered a potential DDoS attack target. An alert is raised if a specific number or percentage of switches are identified as potential targets within a specified time frame.

Algorithm 1 is a The DDoS detection method functions on the basis that each attack node inside a botnet uses standard programming logic that has been pre-established to route network traffic to the target. Thus, attack traffic flows frequently resemble the much more dynamic and varied patterns of real network traffic (Valizadeh and Taghinezhad-Niar, 2022). As a result, attack traffic flows have distinct packet header properties compared to regular traffic flows. The first step in calculating traffic is calculating the actual number of packets arriving and comparing it to the threshold a. Alternatively, comparing the current rate of traffic arriving in each time frame Tw can help detect attacks when there is a noticeable divergence from the data distance value.

If the $P\_i,j/S\_i,j$ value is more than the threshold value, we classify it as attack traffic; if not, we classify it as valid traffic. Although high-rate traffic may be categorized as legitimate traffic, it may also be the consequence of an unanticipated increase in network traffic. We will first establish the threshold value using the recommended detection approach, after which we will utilize algorithm 1 to recognize the attack, algorithm 2 to carry out mitigation, and algorithm 1 to safeguard our data flow.

Algorithm 1: A DDoS attack detection Algorithm

1. Set
    a. M: number of switch ports in SDN network
    b. s_i flow statistics for switch i
    c. p_i,j packet count for flow j on switch i

d. $b\_i\ byte$ count for all flows on switch i
e. T sampling period
f. Tw Time window $size\ = 1\ sec.$
g. a threshold at each switch s_i

2. For each switch port i
3. While $T >= Tw$, examine the network traffic arising from the switches
4. Features that extract packet headers: $F - (srcIP;\ dstIP;\ pktsize;\ no.ofpkts(s\_i))$ classify current Tw into distinct network flows at each s_i.
5. Determine the threshold for each network flow on each switch based on the current Tw at each s_i. using Eq. (1).
6. Compute the average and standard deviation of the threshold for each switch network flow using Eqs. (2-3) respectively at each s_i.
7. Identify threshold value using Eq. (4)
8. **For** $j = 1: M$
9. **if** $s\_i, j\ >\ a$ then
10. Declare the traffic as DDoS
11. **else**
12. **end if**
13. Traffic may be legitimate or LR-DDoS
14. **else**
15. Declare the traffic as Legitimate.
16. **end if**
17. increment Tw and go to step 2.

Algorithm 2: DDoS attack Mitigation

1. Function Mitigation
2. Compile the switches and ports for the current flow
3. Verify that the attack is taking place
4. Collect Port and Switches of attack
5. **If** $current - low\ =\ attack - flow$, then
6. Create and modify a new flow entry using attack parameters
7. Send Switch a new flow entry.
8. **else**
9. avoid action
10. **end if**
11. **end** function

The simulations were conducted in phases, with each packet sampled by sFlow for analysis. All floods were passed from host to host, with no defined mitigation threshold. Despite receiving all floods, the mitigation script from other hosts was activated to identify and stop the attack using the mitigation port algorithm.

Results and Discussion

This section covers utilizing the sFlow instrument for identifying and reducing attacks on all controllers. After that, the results of latency, throughput, and jitter are assessed and a comparison graph is displayed.

The normal traffic flow rate was 2000 packets per second (Anyanwu *et al*., 2023; Rozam and Riasetiawan, 2023) until DDoS attacks began to penetrate the system. The controllers are inundated with a massive amount of ICMP traffic. The network is inundated with up to 8,000 packets per second; the rate has since grown to 10,000 packets per second. The departure from typical traffic and the ICMP faults are readily visible. Figures 3 show typical traffic as red lines, while ICMP errors are blue bars.

The orange line graph represents the threshold line for a rising graph attack, while the blue line shows the tool's ability to detect and secure data flow through mitigation and upgrade. The bandwidth usage in the first 5 sec remains consistent for both scenarios, but a DDoS attack is launched on the 9th sec. The two scenarios showed similar results until the 33 sec mark. The unprotected scenario's bandwidth consumption increased due to lower hard and idle timeout parameters, resulting in 38% more entries in the flow table (Anyanwu *et al*., 2022). The recommended system rejected all attacker machines' packets, preventing the flow table's entries from increasing. This is because, although legitimate hosts also produce traffic, up until that moment, all of it was produced by the attacking servers. We can also observe how the sFlow tool is used through the graphic. After identifying the attack, it employs a mitigation port to limit data flow and safeguard the attacker's host's remaining data.

The sFlow analyzer (h8) observed that traffic volume increased to 10,000 packets per second during a flood attack from attacking sites directed toward the host, indicating the rapid exhaustion of network resources (Sheibani *et al*., 2022). Switches' limited memory leads to the flow table filling up quickly, necessitating that the entire packet be delivered to the controller. A protective mechanism is crucial to prevent network failure without overloading the controller. The sFlow detection and mitigation technique (Swami *et al*., 2023) demonstrated success by setting a polling threshold of 1 sec forwarding one packet per twelve for processing. After setting a handling rule and updating the flow table, traffic crossed the threshold line, mitigating the attack in 3 sec. Dropped packets originated from infected ports.

Table 1 shows that the DDoS detection time grows with the rate of network traffic for the various situations and parameters employed. We have also taken care of other crucial aspects while the DDoS attempts were being detected. The most crucial information is that, in the initial situation, where just 10,000 packets were flooded and a variety of hosts, switches, and topologies were utilized, each attack resulted in a different degree of packet loss. The linear topology had fewer hosts, resulting in higher packet loss.

Table 1: The outcomes of various scenarios with tree topology

Type of attack	Controller	No. of packets /second	Time to identify a DDoS attack (in seconds)	Round Trip Time (RTT) (in seconds)	Packet Loss (%)
ICMP	ODL	10,000	1.78	77.40	86.50
ICMP	ONOS	10,000	4.12	108.45	99.05
ICMP	RYU	10,000	2.40	119.02	100.00
ICMP	POX	10,000	5.01	142.25	97.00
ICMP	NOX	10,000	6.70	286.11	99.00
ICMP	Floodlight	10,000	9.30	178.20	100.00

Table 2: Comparison of jitter, throughput, and latency value in sequence for SDN controllers

Controllers	Jitter (ms)	Throughput (rates/Gbits/s)	Latency (ms)
ODL	0.14	5.39	2.204
ONOS	0.94	8.57	4.968
RYU	0.53	7.96	7.307
POX	0.43	3.02	13.790
NOX	0.89	4.87	109.149
Floodlight	0.87	11.01	19.550

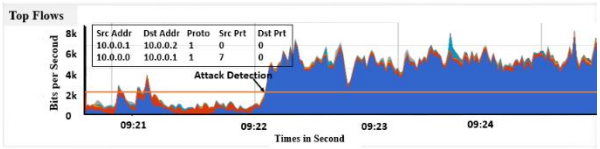

Fig. 3: After a flood of attacks on the host and the detection of the attack by sFlow

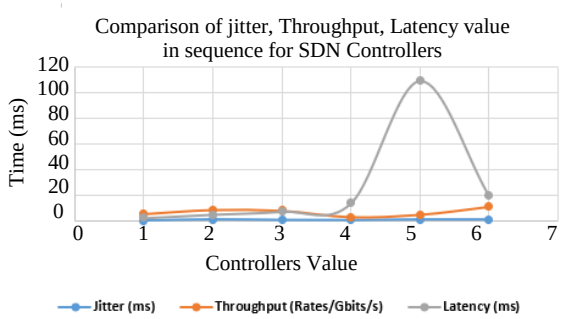

Fig. 4: Comparison of jitter, throughput, and latency value for controllers

The latency, ping delays, jitter, and throughput of the networks were assessed during all trials for all controllers evaluated in various standalone test configurations, in Table 2. Given that a controller needs some time to run tests for different traffic types and packet sizes, we employed IP traffic with ICMP messages of various sizes to measure performance (Sai *et al*., 2022). The performance parameters throughput, jitter, and latency were recorded for each run. The first ping delay was utilized to analyze the flow of data between the hosts that are conceptually distinct in each topology. The throughput is determined by executing the iperf statement (Sritharan *et al*., 2022) in the emulation Command Line Interface (CLI) console, which gives the bandwidth. The final batch of ten ICMP messages was collected in order to calculate jitter. When the controller implements our recommended procedure, there are frequent instances of a sudden increase in CPU utilization. This is because, when the table is huge, our approach tries to get the mean value of the table, which causes the CPU to utilize 100% of its resources. Having said that, this is only immediate.

The table above shows that the ODL controller consistently has slower average ping times than the remaining controllers. In the event of an ICMP DDoS attack, the range for the actual ping time is significantly less. In comparison to the ODL controller, other controller latency performance indicates that it is a less effective controller. As a result, the average RTT of the ONOS, POX, NOX, Floodlight, and Ryu controllers is lower than that of the ODL controller. These observations were made primarily because the remaining controllers employed hybrid commercial tactics, while ODL placed a stronger emphasis on data centers. Based on the test findings, it can be concluded that some controllers perform well in terms of jitter.

According to Fig. 4, the ODL-connected topologies have significantly higher ICMP throughput than the other controllers-connected networks. This is probably due to the fact that it already supports very large-scale networks. The measurements in the virtual test environment are totally reliant on the capability of the controller. This enables us to confirm that ODL offers greater performance in terms of throughput.

The research assessed the performance of SDN-based networks (Shah *et al*., 2022) using IP traffic with ICMP packets. According to the selected criteria, it was discovered that ODL performs better and logs off before ONOS, POX, NOX, Floodlight, and Ryu controllers. The DDoS attacks were launched between hosts, each with its own independent variable. The study used an emulator-ODL, ONOS, POX, NOX, Floodlight, and Ryu controller and sFlow testbed to assess SDN functionality. It compared attack types and access points with the latency, jitter, and throughput (dependent variables) of SDN network controllers in a large scenario. The effect was the dependent variable and changes in the independent variable affected its value.

Conclusion

The research assessed the performance of SDN-based networks using IP traffic with ICMP packets. A

DDoS attack was launched between hosts using emulation analysis and sFlow tools. A virtual traffic flow was generated during the experiment setup to simplify performance measurements. The study evaluated SDN functionality using an emulator- ODL, ONOS, POX, NOX, Floodlight, and Ryu controller and sFlow testbed. An ODL controller-based SDN network was used to imitate a monitoring system that operates in real-time and is capable of detecting DDoS flood attacks. This study evaluates SDN network (Shakil *et al*., 2022) controllers' latency, jitter, and throughput in a large scenario, paired with attack type and access points. ODL outperforms ONOS, POX, NOX, Floodlight, and Ryu controllers in performance but requires specific memory, making it the better choice due to Python-based controllers. This study can help different academics and industrialists choose between these controllers in a variety of application scenarios, such as servers and the Web of things. This research effort opens a lot of other research possibilities. To have a complete image showing these controllers' performance evaluation, we intend to continue expanding this study with more southbound and northbound APIs and clustering many controllers.

Acknowledgment

I would like to express my sincere gratitude to my guide, Dr. Sarvesh Tanwar, and Dr. Sumit Badotra, for their invaluable guidance, unwavering support, and expert mentorship throughout the entire research process.

Funding Information

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author's Contributions

Neelam Gupta: Collected the data and identify the gaps: Assess the controllers' performance. To the best of our knowledge, there aren't many studies in the literature that concentrate on actively measuring the performance of SDN controllers.

Conceived and designed the analysis: The experiment design deviates from the methods used in this publication and the research only takes a few topologies into account.

Contributed data or analysis tools: The SDN controllers work in different DDoS attack situations.

Performed the analysis: The result of throughput, jitter, and latency is depicted inside comparative results.

Wrote the manuscript: It is crucial to assess new releases of these controllers to better comprehend the performance enhancements.

Sarvesh Tanwar: Help draft the manuscript, provided valuable guidance and mentorship throughout the research process, and ensured the paper and quality and rigor.

Sumit Badotra: The author provided expert guidance, critical insights, and mentorship throughout the research process ensuring the paper and quality and rigor.

Ethics

The corresponding author declared that this study has not been submitted elsewhere.

References

Alhijawi, B., Almajali, S., Elgala, H., Salameh, H. B., & Ayyash, M. (2022). A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets. *Computers and Electrical Engineering*, *99*, 107706. https://doi.org/10.1016/j.compeleceng.2022.107706

Ali, M. N., Imran, M., din, M. S. U., & Kim, B. S. (2023). Low rate DDoS detection using weighted federated learning in SDN control plane in IoT network. *Applied Sciences*, *13*(3), 1431. https://doi.org/10.3390/app13031431

Anyanwu, G. O., Nwakanma, C. I., Lee, J. M., & Kim, D. S. (2022). Optimization of RBF-SVM kernel using grid search algorithm for DDoS attack detection in SDN-based VANET. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2022.3199712

Anyanwu, G. O., Nwakanma, C. I., Lee, J. M., & Kim, D. S. (2023). RBF-SVM kernel-based model for detecting DDoS attacks in SDN integrated vehicular network. *Ad Hoc Networks*, *140*, 103026. https://doi.org/10.1016/j.adhoc.2022.103026

Batool, S., Zeeshan Khan, F., Qaiser Ali Shah, S., Ahmed, M., Alroobaea, R., Baqasah, A. M., ... & Ahsan Raza, M. (2022). Lightweight statistical approach towards TCP SYN flood DDOS attack detection and mitigation in SDN environment. *Security and Communication Networks*, *2022*. https://doi.org/10.1155/2022/2593672

Gupta, N., Maashi, M. S., Tanwar, S., Badotra, S., Aljebreen, M., & Bharany, S. (2022a). A comparative study of software defined networking controllers using mininet. *Electronics*, *11*(17), 2715. https://doi.org/10.3390/electronics11172715

Gupta, N., Tanwar, S., Badotra, S., & Behal, S. (2022b). Performance Analysis of SDN controller. *International Journal of Performability Engineering*, *18*(8), 537. https://doi.org/10.23940/ijpe.22.08.p1.537544

Gupta, N., Tanwar, S., & Badotra, S. (2023). Review of Software-Defined Network-Enabled Security. *In Computational Intelligence for Engineering and Management Applications: Select Proceedings of CIEMA 2022* (pp. 427-441). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-8493-8_33

Jiang, S., Yang, L., Gao, X., Zhou, Y., Feng, T., Song, Y., ... & Cheng, G. (2022). Bsd-guard: A collaborative blockchain-based approach for detection and mitigation of SDN-targeted DDOS attacks. *Security and Communication Networks*, *2022*, 1-16. https://doi.org/10.1155/2022/1608689

Kumar, P., Baliyan, A., Prasad, K. R., Sreekanth, N., Jawarkar, P., Roy, V., & Amoatey, E. T. (2022). Machine learning enabled techniques for protecting wireless sensor networks by estimating attack prevalence and device deployment strategy for 5G networks. *Wireless Communications and Mobile Computing*, *2022*. https://doi.org/10.1155/2022/5713092

Mehra, A., & Badotra, S. (2022). A Novel Framework for Prevention against DDoS Attacks using Software Defined Machine Learning Model. *International Journal of Performability Engineering*, *18*(8), 580. https://doi.org/10.23940/ijpe.22.08.p6.580588

Mishra, A., & Gupta, N. (2022). Supervised Machine Learning Algorithms Based on Classification for Detection of Distributed Denial of Service Attacks in SDN-Enabled Cloud Computing. In *Cyber Security, Privacy and Networking: Proceedings of ICSPN 2021* (pp. 165-174). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-16-8664-1_15

Prasad, S., Prasad, A., Arockiasamy, K., & Yuan, X. (2022, February). Emulation and Analysis of Software-Defined Networks for the Detection of DDoS Attacks. In *International Conference on Computer, Communication and Signal Processing*, (pp. 213-231). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-11633-9_16

Rozam, N. F., & Riasetiawan, M. (2023). XGBoost Classifier for DDOS Attack Detection in Software Defined Network Using sFlow Protocol. *International Journal on Advanced Science, Engineering and Information Technology*, *13*(2). https://doi.org/10.18517/ijaseit.13.2.17810

Sai, A. D., Tilak, B. H., Sanjith, N. S., Suhas, P., & Sanjeetha, R. (2022, October). Detection and Mitigation of Low and Slow DDoS attack in an SDN environment. In *2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, (pp. 106-111). IEEE. https://doi.org/10.1109/DISCOVER55800.2022.9974724

Shah, S. Q. A., Khan, F. Z., & Ahmad, M. (2022). Mitigating TCP SYN flooding based EDOS attack in cloud computing environment using binomial distribution in SDN. *Computer Communications*, *182*, 198-211. https://doi.org/10.1016/j.comcom.2021.11.008

Shakil, M., Fuad Yousif Mohammed, A., Arul, R., Bashir, A. K., & Choi, J. K. (2022). A novel dynamic framework to detect DDoS in SDN using metaheuristic clustering. *Transactions on Emerging Telecommunications Technologies*, *33*(3), e3622. https://doi.org/10.1002/ett.3622

Sheibani, M., Konur, S., & Awan, I. (2022, August). DDoS Attack Detection and Mitigation in Software-Defined Networking-Based 5G Mobile Networks with Multiple Controllers. *In 2022 9th International Conference on Future Internet of Things and Cloud* (*FiCloud*), (pp. 32-39). IEEE. https://doi.org/10.1109/FiCloud57274.2022.00012

Sritharan, K., Elagumeeharan, R., Nakkeeran, S., Mohamed, A., Ganegoda, B., & Yapa, K. (2022, October). Machine Learning Based Distributed Denial-of-Services Attacks Detection and Mitigation Testbed for SDN-Enabled IoT Devices. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, (pp. 1-6). IEEE. https://doi.org/10.1109/ICCCNT54827.2022.9984248

Swami, R., Dave, M., & Ranga, V. (2023). IQR-based approach for DDoS detection and mitigation in SDN. *Defence Technology*, *25*, 76-87. https://doi.org/10.1016/j.dt.2022.10.006

Tao, H. Y., Huang, C. K., & Shen, S. H. (2023, July). A Low-overhead Network Monitoring for SDN-Based Edge Computing. *In 2023 IEEE Symposium on Computers and Communications (ISCC)*, (pp. 600-606). IEEE. https://doi.org/10.1109/ISCC58397.2023.10218002

Udhaya Prasath, M., Sriram, B., Prakashkumar, P., & Vetriselvi, V. (2022). DDoS mitigation in SDN using MTD and behavior-based forwarding. In *Innovations in Electronics and Communication Engineering: Proceedings of the 9th ICIECE 2021* (pp. 373-380). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-16-8512-5_40

Valizadeh, P., & Taghinezhad-Niar, A. (2022, May). Ddos attacks detection in multi-controller based software defined network. *In 2022 8th International Conference on Web Research (ICWR)*, (pp. 34-39). IEEE. https://doi.org/10.1109/ICWR54782.2022.9786246

Yungaicela-Naula, N. M., Vargas-Rosales, C., Pérez-Díaz, J. A., & Carrera, D. F. (2022). A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning. *Journal of Network and Computer Applications*, *205*, 103444. https://doi.org/10.1016/j.jnca.2022.10344