Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.042432

ARTICLE

Data Analysis of Network Parameters for Secure Implementations of SDN-Based Firewall

Rizwan Iqbal[1,*], Rashid Hussain[2], Sheeraz Arif[3], Nadia Mustaqim Ansari[4] and Tayyab Ahmed Shaikh[2]

[1]Department of Telecommunication Engineering, Dawood University of Engineering & Technology, Karachi, Pakistan

[2]Faculty of Engineering Sciences and Technology, Hamdard University, Karachi, Pakistan

[3]Faculty of Information Technology, Salim Habib University, Karachi, Pakistan

[4]Department of Electronic Engineering, Dawood University of Engineering & Technology, Karachi, Pakistan

*Corresponding Author: Rizwan Iqbal. Email: rizwan.iqbal@duet.edu.pk

Received: 30 May 2023 Accepted: 29 August 2023 Published: 29 November 2023

ABSTRACT

Software-Defined Networking (SDN) is a new network technology that uses programming to complement the data plane with a control plane. To enable safe connection, however, numerous security challenges must be addressed. Flooding attacks have been one of the most prominent risks on the internet for decades, and they are now becoming challenging difficulties in SDN networks. To solve these challenges, we proposed a unique firewall application built on multiple levels of packet filtering to provide a flooding attack prevention system and a layer-based packet detection system. This study offers a systematic strategy for wrapping up the examination of SDN operations. The Mininet simulator examines the effectiveness of SDN-based firewalls at various network tiers. The fundamental network characteristics that specify how SDN should operate. The three main analytical measures of the network are jitter, response time, and throughput. During regular operations, their behavior evaluates in the standard SDN conditions of Transmission Control Protocol (TCP) flooding and User Datagram Protocol (UDP) flooding with no SDN occurrences. Low Orbit Ion Cannon (LOIC) is applied to launch attacks on the transmission by the allocated server. Wireshark and MATLAB are used for the behavioral study to determine how sensitive the parameters are used in the SDN network and monitor the fluctuations of those parameters for different simulated scenarios.

KEYWORDS

Software defined networking; firewall; POX controller; open v switch; Mininet; OpenFlow

1 Introduction

Due to several claims that rely on data-driven and reliable facts-sharing paradigms, data networks are becoming more prevalent. The Industrial Internet of Things (IIoT) and the Internet of Things (IoT) are the most dominant. As they develop and are exposed to increasingly unequal dynamic behavioral changes, current network architectures still have a management complexity problem. Software-Defined Networking (SDN) addresses the difficulty of managing contemporary data networks [1,2].

Recent studies have examined several significant new technologies, including edge/fog computing, Blockchain (BC), SDN, IIoT, 5G, Machine Learning (ML), and Wireless Sensor Networks (WSN).

All these technologies offer data confidentiality, integrity, and authenticity in their respective use cases (particularly in the business sector). SDN provides infrastructure flexibility, while BC safeguards the privacy of smart IIOT services [3].

SDNs perform separate control, management, and forwarding planes [4]. To aid with this endeavor, SDNs enable openness. Most modern data networks are open, programmable, and adaptable. In a typical SDN, operators can dynamically configure network—device programs to ensure the best possible resource distribution and usage. The SDN controller stores all network information like topology, protocols, applications, and security parameters. This architecture permits forwarding devices in SDNs to implement an integrated interface for data interchange with the SDN controller. They obtain the operational state of the network for network operations such as traffic engineering. Giving to the literature, the critical issue in modern SDN-based architecture with the enlarged coding, flexible environment, and scalable like IoT and IIoT networks. All mentioned reasons increased vulnerability and compromised security results in the Distributed Denial of Service (DDoS) flooding attacks [5].

Conventionally, DDoS attacks are well-planned attacks from several hacked hosts. They are directed at network nodes or end-user devices to steal their share of the available bandwidth or entirely shut them down. Researchers have launched DDoS flooding attacks in previous years. They are considered the most hazardous means of traffic on the internet. Hence, these flooding attacks are well investigated in the current body of literature [6,7]. According to numerous researchers in this field, there is no standard technique to handle these types of attacks. Typically, attackers use a botnet to launch attacks. Consequently, device node operators on attacked networks frequently ignore that attacks originate from their hardware and network addresses [8].

Modern SDNs are being researched by analyzing data network performance indicators and the many types of DDoS flooding attacks. As discussed further in Section 2, it remains a relatively grey area [9–11]. Throughput, jitter, and response time measurements have been used in several research projects to assess the effectiveness and performance of SDNs. They are, therefore, maybe the most well-known SDN performance measures. Network administrators can identify and differentiate DDoS attacks on SDN platforms by analyzing the behavior of SDN performance indicators under common SDN DDoS states.

Consequently, more effective mitigation strategies, action plans, and response plans are devised to ensure the consistency and accessibility of current and expectations of SDNs. This study uses the emulated network's jitter, latency, and throughput to assess how DDoS flooding attack scenarios affect SDN performance [12–14].

A firewall monitors network traffic and decides whether to allow or deny traffic based on predefined rules. Through time, it has provided network security. It also analyses every outgoing and incoming message and blocks those not complying with a predetermined set of restrictions. It aids in overcoming issues caused by attackers by employing various forms of attacks [15]. Types of firewalls include Stateful and Stateless firewalls. Packets that fit into tracked links can be passed through a stateful firewall using attributes such as source and destination IP addresses, port numbers, and sequence numbers. Identify the communication path during the transfer of the packet. Packet encryption is also an option [16]. It is safer than a stateless firewall regarding security and throughput for significant traffic volumes. Similarly, using its ruleset, a stateless firewall verifies and forwards all incoming packets. By using distinct dynamic criteria, machine learning algorithms will improve the capacity of security solutions for protecting the upcoming 5G and future networks [17].

The SDN model demonstrated it in the literature, and the related dataset was analyzed and evaluated by a representative of actual SDN and SDN scenarios. As a result, additional research was conducted on the SDN model, and different data was extracted to describe the findings by evaluating the performance parameters of the SDN network through TCP and UDP protocols in the presence of DDoS attacks. Jitter, latency, and throughput are common SDN performance indicators that have been the focus of behavioral research on SDN operations. Reliable SDN performance indicators, including jitter, response time, and throughput, were subjected to regression-based sensitivity analysis (RSA). Determine their pairwise relationships for practical SDN operations. Low-cost, inference-driven characterization and evaluation of SDN processes in practice [18–20].

Security is a major problem because it was not adequately considered in the original design of the SDN architecture. Therefore, it is crucial to construct a reliable security mechanism to safeguard the network from both inside and outside threats without compromising the SDN's intended function. To put it another way, defense-in-depth is typically regarded as a good solution for networks with varying degrees of trust. Authors who aimed to improve network safety were inspired to work with SDN technology because of its logical and centralized management.

2 Related Work

DDoS attacks launch against an SDN network's infrastructure, control, and application layers; the consequences can be significantly more severe than in conventional networks. Therefore, for the immediate or real-time detection and evaluation of problematic SDN activities, it is necessary to conduct an interactive analysis of the performance metrics that serve as the foundation for the proper operation of SDN. Network performance measurements typically investigate the potential quality of service enhancement strategies in SDN environments. Examples include anticipating packet scheduler actions for hierarchical token buckets (HTB), stochastic fairness queueing (SFQ), and random early detection using different metrics values like response time and throughput of the SDN. Consequently, the HTB and SFQ packet scheduler was superior to the other evaluated typical response time and throughput designs. Even though the work accurately illustrates a specific fraction of SDN for quality-of-service monitoring, a comprehensive behavioral investigation of the SDN metrics has not been done [21,22]. According to the researcher's analysis, new types of attacks occurred in cloud systems, such as flow table overloading DDoS attacks. Flow table sharing is the proposed method to encounter defined attacks [23]. There are many possibilities to reduce the effect of DDoS attacks in cloud computing, but DDoS attacks are still creating problems for researchers and host resources. This paper proposed a low-overhead effective operation for handling DDoS attacks. The proposed mechanism is based on the entropy value between DDoS attacks and normal traffic [24]. Researchers intended a protective activity to detect such attacks by analyzing entropy values. Then, according to entropy values, the SDN-cloud-enabled online social network (OSN) will block that port to combat DDoS attacks [25].

Performance parameters like jitter, latency, throughput, and packet loss investigate to enhance the SDN network's quality. In every instance, the QoS-aware routing method (QRS) suggested and chose a route with minimum jitter, end-to-end delay, and greater throughput [26]. However, illative evaluations of how SDN performance metrics change between states do not outperform experimental designs. We utilized Mininet and GNS3 to simulate and characterize both an SDN and a traditional network. It is important to compare the metrics used to measure the stability of SDN designs with those used to measure the strength of conventional network topologies. No behavioral evaluation of network performance measures [27] was provided, even though experimental latency analysis showed that the SDN provides a mean latency around three times lower than the legacy system.

The researchers have reported a local sensitivity study of throughput, reaction time, and jitter measurements in different SDN conditions. Due to widespread DDoS flooding attacks against the SDN, the SDN's throughput, jitter, and reaction time measurements are statistically sensitive to the transition from normal to abnormal operations. Jitter is the most sensitive SDN metric or parameter. As a result, there is a need for a comprehensive analytical approach that can assess the whole distributions and tendencies of SDN parameters and metrics. This study follow-up confirmed that SDN states effectively map to their significant performance metrics (numerical or continuous variables). All the classifiers we looked at to determine SDN states [28,29].

Depending on specified rules, a firewall monitors network traffic and decides whether to allow or deny traffic. Over time, it has offered network security. Additionally, it looks at each incoming and outgoing message to prohibit any that do not adhere to a set of rules or standards. Employing various attacks aids in solving the issues brought forth by the attackers [30]. Two categories of firewalls exist: stateful and stateless. Attributes of packets, such as source and destination IP addresses, port numbers, and sequence numbers, are utilized within tracked connections. A stateful firewall monitors the link. It identifies the communication path during packet transmission. Encrypting messages is another method [31].

It is more secure than a stateless firewall regarding security and speed for considerable traffic volumes. In a stateless firewall, all incoming packets are similarly examined and forwarded according to the ruleset. The link never traces, as opposed to a stateful firewall. Based on source and destination IP addresses, network traffic is analyzed without knowledge of the traffic pattern or data flow [32].

The architecture of ML-based intrusion detection systems (IDSs) for a better comprehension of recent forms of intrusion detection in SDNs and to provide major hints for future research in this area [33]. Machine learning methods can strengthen detection rates, decline false alarm rates, and realistic processing and transmission values [34].

The Graph connectivity method and several trust-based routing principles are used for Crossfire attack detection and mitigation. The norms are applied in the SDN switches for better bandwidth utilization [35].

Based on the works, most current works in accessible literature provide solutions for boosting the quality of service to detect and categorize the status of the SDN in the face of attacks. To illustrate and draw inferences regarding the SDN's behavior under stress, the author employs exploratory data analysis and regression-based sensitivity analysis, such as DDoS attacks. It offers a new paradigm and understanding for safeguarding the SDN in the face of attacks like DDoS.

3 Methodology and Implementations

This paper discusses the implementation of a firewall on the SDN POX controller that admits or rejects data traffic, including the MAC address, IP address, and port address. As shown in Fig. 1, SDN functional structure with a firewall based on the application, control, and data layer. The application layer consists of different applications like routing, security, and others. The control layer manages TCP and UDP traffic flows in real-time. After receiving the data flow, the payload decided their route from source to destination. The information regarding network connectivity, like IP and port addresses of incoming and outgoing packets, session management and packet sequence management examined to establish the connection. Firewall rules are applied to the traffic flow by using a state table. The occurrence of every event because of the firewall rule is recorded and updated in the session table.

Flow-based scheduling decreases path selection method and bandwidth surety for flows and expands scalability. The data layer uses OpenFlow switches to create topologies.

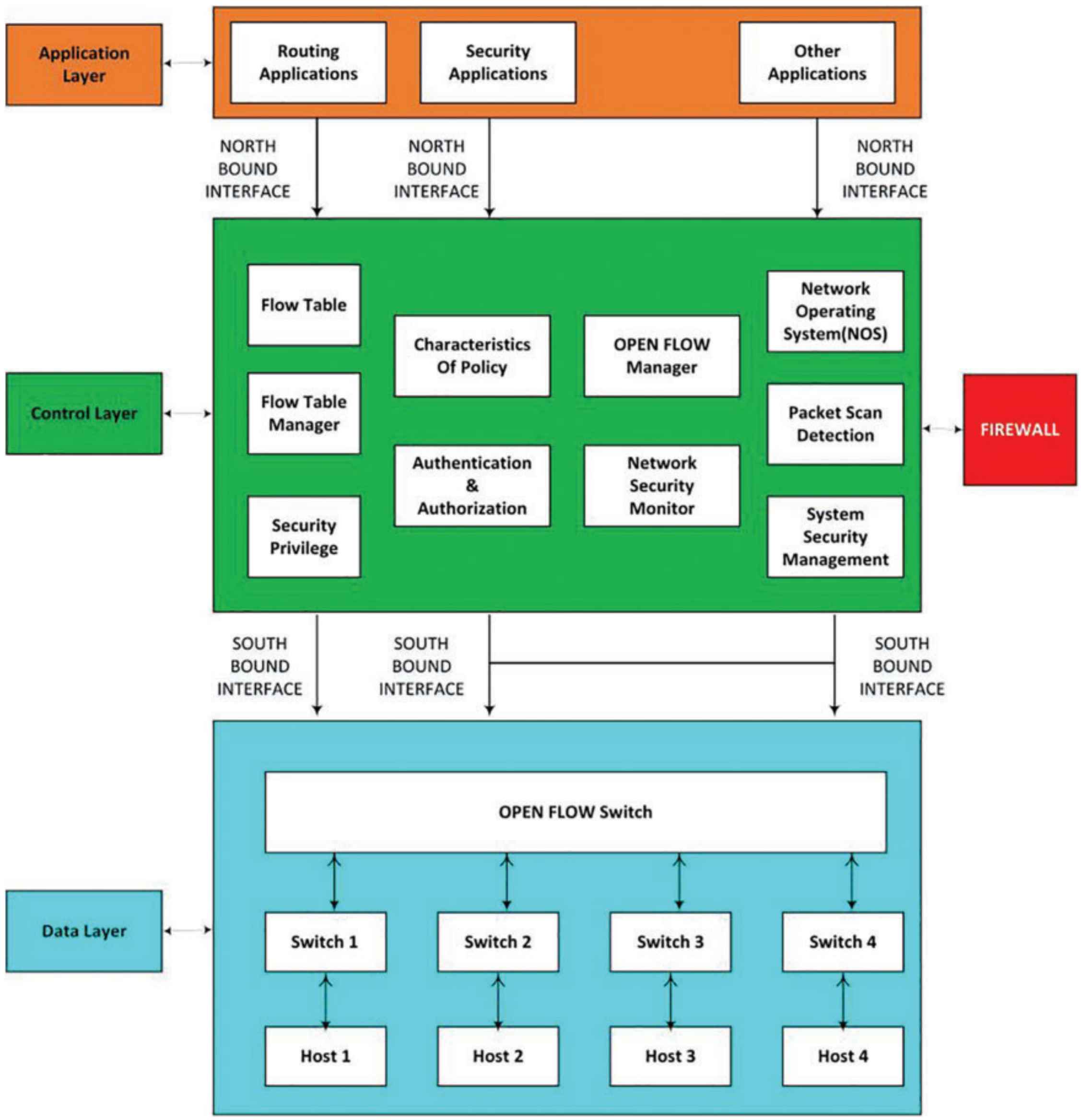


Figure 1: SDN functional structure with firewall [30]

Create SDN-based custom topology using open v switch to implement firewall rules on network layers 2, 3, and 4. TCP and UDP scenarios are used to validate the firewall rules. After that, evaluate the system by measuring the performance of network parameters like round trip time, bandwidth utilization, latency, and data loss. To check the proposed firewall in the normal and under flooding attack scenario with throughput, response time, and jitter values. LOIC is applied to launch attacks on the transmission by the allocated server. The efficiency of the recommended network was determined by measuring the network's throughput, latency, and response time. Comparing the performance results revealed the firewall-enabled controller as an SDN network environment with great potential, as shown in Fig. 2.

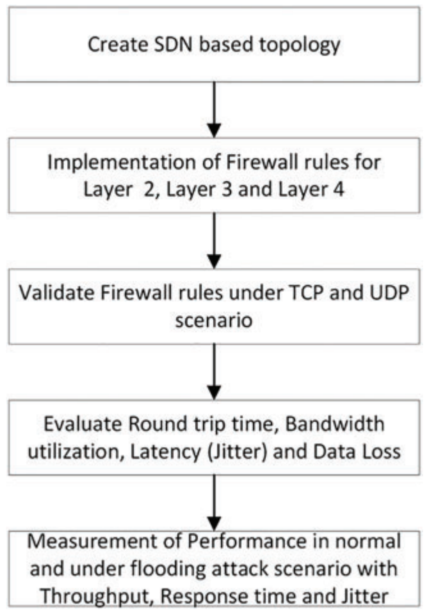


Figure 2: Flow graph of the proposed system

As depicted in Fig. 3, the custom topology employs a 180 GB HDD, 8 GB RAM, 2.4 GHz processor, and Linux as the operating system. On Ubuntu 16.04 LTS, the Mininet emulator 2.3.0 and the POX controller are installed. The experiment design Python considers nine OpenFlow switches and sixty-four associated hosts or end nodes to define topology and firewall rules.

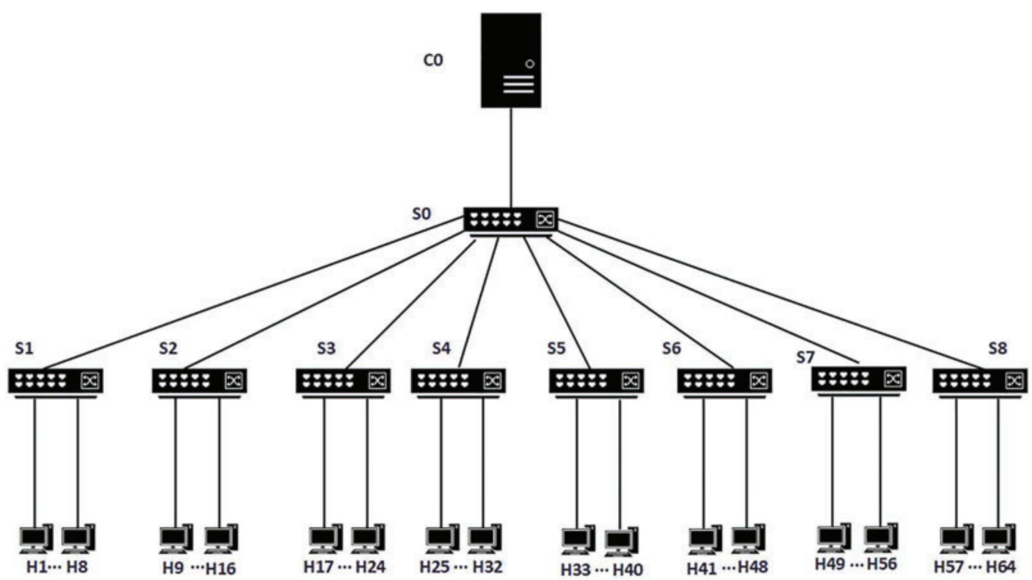


Figure 3: SDN-based topology

Table 1 shows the configuration of IP addresses, MAC addresses, and port numbers of the controller, switches, and hosts.

Table 1: Configuration of network nodes

Device name	IP address	MAC address
POX controller	127.0.0.1	Port no. 6636
Host h1	10.0.0.1	00:00:00:00:00:01
To	To	To
Host h64	10.0.0.64	00:00:00:00:00:40

The following procedures are used to generate SDN data with custom firewall rules. Table 2 shows firewall rules for layer 2.

Table 2: Layer 2 firewall rules for multiple devices

Device	Source MAC address	Destination	Action
h2	00:00:00:00:00:02	Any	Deny
h12	00:00:00:00:00:0c	Any	Deny
h22	00:00:00:00:00:16	Any	Deny
h32	00:00:00:00:00:20	Any	Deny
h42	00:00:00:00:00:2a	Any	Deny
h52	00:00:00:00:00:34	Any	Deny
h62	00:00:00:00:00:3e	Any	Deny
h8	00:00:00:00:00:08	Any	Deny
h18	00:00:00:00:00:12	Any	Deny
h28	00:00:00:00:00:1c	Any	Deny
h38	00:00:00:00:00:26	Any	Deny
h48	00:00:00:00:00:30	Any	Deny
h58	00:00:00:00:00:3a	Any	Deny
h14	00:00:00:00:00:0e	Any	Deny
h25	00:00:00:00:00:19	Any	Deny
h34	00:00:00:00:00:22	Any	Deny
h44	00:00:00:00:00:2c	Any	Deny
h54	00:00:00:00:00:36	Any	Deny
h64	00:00:00:00:00:40	Any	Deny
h15	00:00:00:00:00:0f	Any	Deny

Table 3 shows firewall rules for layer 3.

Table 4 shows firewall rules for layer 4.

Verify network connectivity across hosts using the "ping" command. For example, host 2 established a connection with host 4 by implementing firewall rules from Table 2, as shown in Fig. 4.

Table 3: Layer 3 firewall rules for multiple devices

Source device	Source IP address	Destination device	Destination IP address	Action
h1	10.0.0.1	h4	10.0.0.4	Deny
h11	10.0.0.11	h13	10.0.0.13	Deny
h21	10.0.0.21	h23	10.0.0.23	Deny
h31	10.0.0.31	h33	10.0.0.33	Deny
h41	10.0.0.41	h43	10.0.0.43	Deny
h51	10.0.0.51	h53	10.0.0.53	Deny
h61	10.0.0.61	h63	10.0.0.63	Deny
h6	10.0.0.6	h9	10.0.0.9	Deny
h16	10.0.0.16	h19	10.0.0.19	Deny
h26	10.0.0.26	h29	10.0.0.29	Deny
h36	10.0.0.36	h39	10.0.0.39	Deny
h46	10.0.0.46	h49	10.0.0.49	Deny
h56	10.0.0.56	h59	10.0.0.59	Deny
h13	10.0.0.13	h21	10.0.0.21	Deny
h23	10.0.0.23	h31	10.0.0.31	Deny
h33	10.0.0.33	h41	10.0.0.41	Deny
h43	10.0.0.43	h51	10.0.0.51	Deny
h53	10.0.0.53	h61	10.0.0.61	Deny
h63	10.0.0.63	h1	10.0.0.1	Deny

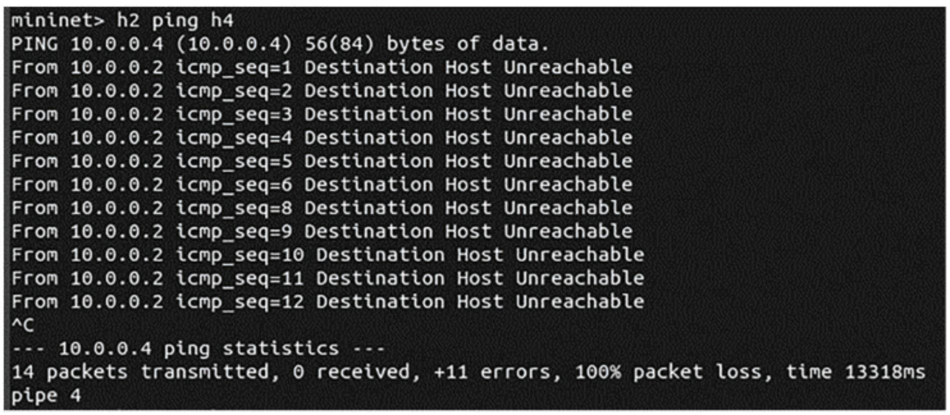
Table 4: Layer 4 firewall rules for multiple devices

Source	Destination device	Destination server IP address	Port number	Action
Any	h3	10.0.0.3	80	Deny
Any	h5	10.0.0.5	80	Deny
Any	h15	10.0.0.15	80	Deny
Any	h25	10.0.0.25	80	Deny
Any	h35	10.0.0.35	80	Deny
Any	h45	10.0.0.45	80	Deny
Any	h55	10.0.0.55	80	Deny
Any	h7	10.0.0.7	80	Deny
Any	h17	10.0.0.17	80	Deny
Any	h27	10.0.0.27	80	Deny
Any	h37	10.0.0.37	80	Deny
Any	h47	10.0.0.47	80	Deny
Any	h57	10.0.0.57	80	Deny
Any	h10	10.0.0.10	80	Deny
Any	h20	10.0.0.20	80	Deny

(Continued)

Table 4 (continued)

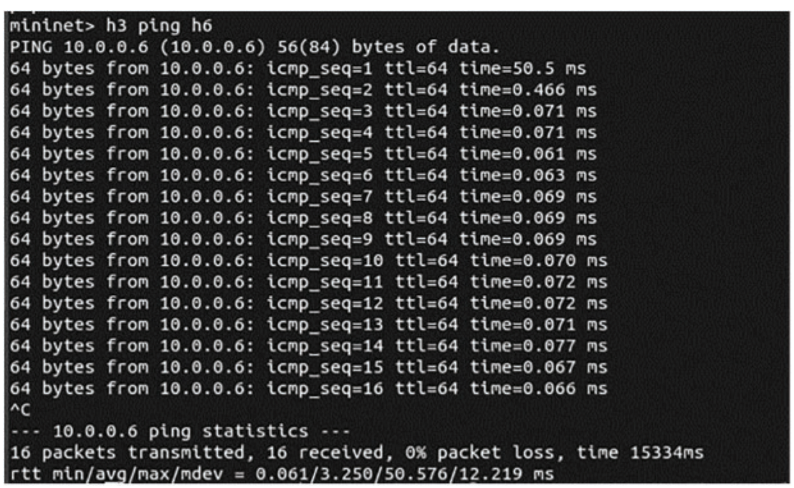
Source	Destination device	Destination server IP address	Port number	Action
Any	h30	10.0.0.30	80	Deny
Any	h40	10.0.0.40	80	Deny
Any	h50	10.0.0.50	80	Deny
Any	h60	10.0.0.60	80	Deny



```
mininet> h2 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
From 10.0.0.2 icmp_seq=4 Destination Host Unreachable
From 10.0.0.2 icmp_seq=5 Destination Host Unreachable
From 10.0.0.2 icmp_seq=6 Destination Host Unreachable
From 10.0.0.2 icmp_seq=8 Destination Host Unreachable
From 10.0.0.2 icmp_seq=9 Destination Host Unreachable
From 10.0.0.2 icmp_seq=10 Destination Host Unreachable
From 10.0.0.2 icmp_seq=11 Destination Host Unreachable
From 10.0.0.2 icmp_seq=12 Destination Host Unreachable
^C
--- 10.0.0.4 ping statistics ---
14 packets transmitted, 0 received, +11 errors, 100% packet loss, time 13318ms
pipe 4
```

Figure 4: Connectivity between h2 and h4

In the above scenario, host 2 blocks all traffic for all hosts. So, when the h2 ping h4 command runs, no packets are received, and there is 100% packet loss. Similarly, when pinging h6 from h3, 16 containers were transmitted, and all 16 packages were obtained with 0% packet loss, as shown in Fig. 5.



```
mininet> h3 ping h6
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=50.5 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=64 time=0.466 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 10.0.0.6: icmp_seq=5 ttl=64 time=0.061 ms
64 bytes from 10.0.0.6: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.0.0.6: icmp_seq=7 ttl=64 time=0.069 ms
64 bytes from 10.0.0.6: icmp_seq=8 ttl=64 time=0.069 ms
64 bytes from 10.0.0.6: icmp_seq=9 ttl=64 time=0.069 ms
64 bytes from 10.0.0.6: icmp_seq=10 ttl=64 time=0.070 ms
64 bytes from 10.0.0.6: icmp_seq=11 ttl=64 time=0.072 ms
64 bytes from 10.0.0.6: icmp_seq=12 ttl=64 time=0.072 ms
64 bytes from 10.0.0.6: icmp_seq=13 ttl=64 time=0.071 ms
64 bytes from 10.0.0.6: icmp_seq=14 ttl=64 time=0.077 ms
64 bytes from 10.0.0.6: icmp_seq=15 ttl=64 time=0.067 ms
64 bytes from 10.0.0.6: icmp_seq=16 ttl=64 time=0.066 ms
^C
--- 10.0.0.6 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15334ms
rtt min/avg/max/mdev = 0.061/3.250/50.576/12.219 ms
```

Figure 5: Connectivity between h3 and h6

It concludes that there is connectivity between h3 and h6 and no connectivity between h2 and h4. So, the POX controller blocks only the link for the desired hosts by implementing the firewall on layer 2.

1.Using the “iperf” command, make UDP and TCP servers that will listen on various network ports.

For example, Fig. 6a shows node h3 as a TCP server, and Fig. 6b shows hosts h4, h5, h6, h7, h10, h11, h16, and h17 as a client. According to the predefined rule mentioned in table no. 4, port no. 80 of h3 (10.0.0.3) block for all other devices. In Fig. 6b, all clients want to access port no. 80 of h3 (10.0.0.3), but the port is blocked for all users; there is no transmission, which is confirmed by Fig. 6a. The server cannot listen to any transmission request on port no. 80. All traffic destined to process h3 at port 80 is blocked. No data was sent on port 80 by the server h3.

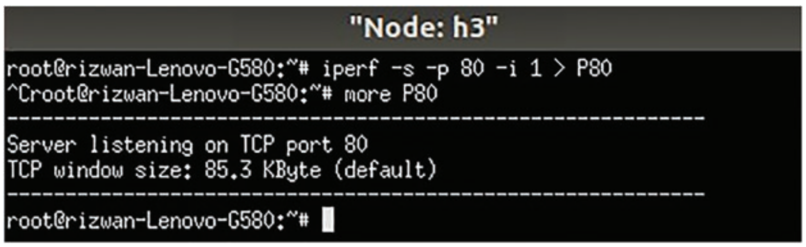


Figure 6a: h3 working as a server

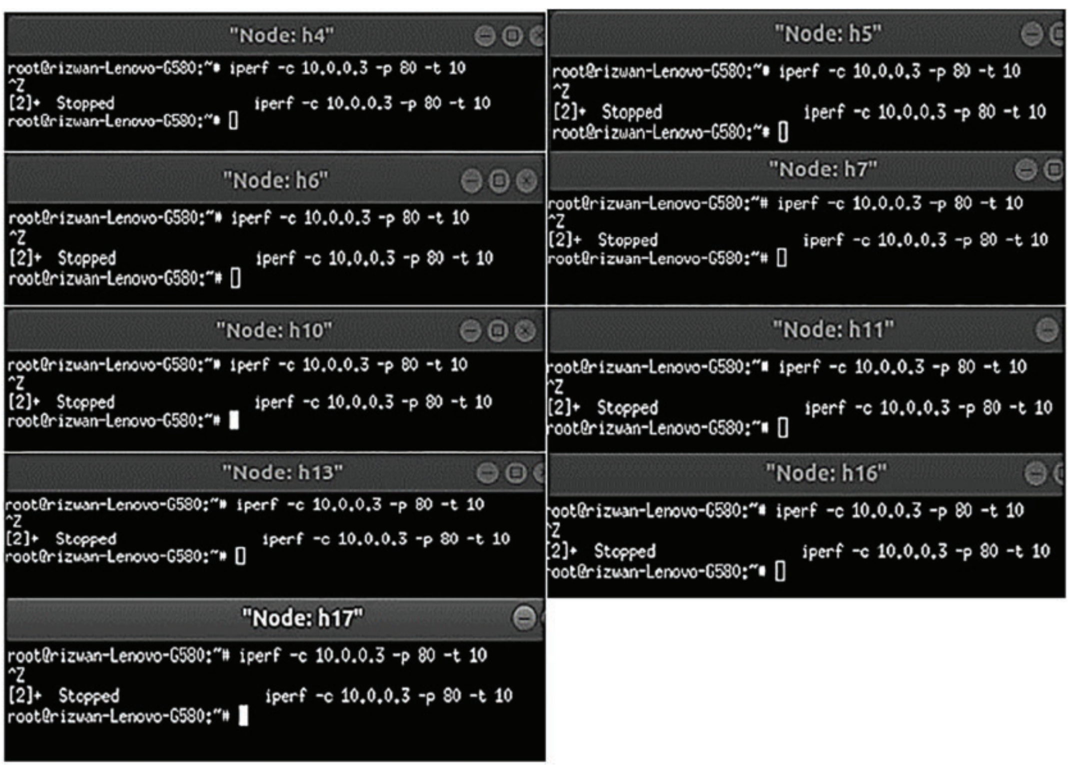
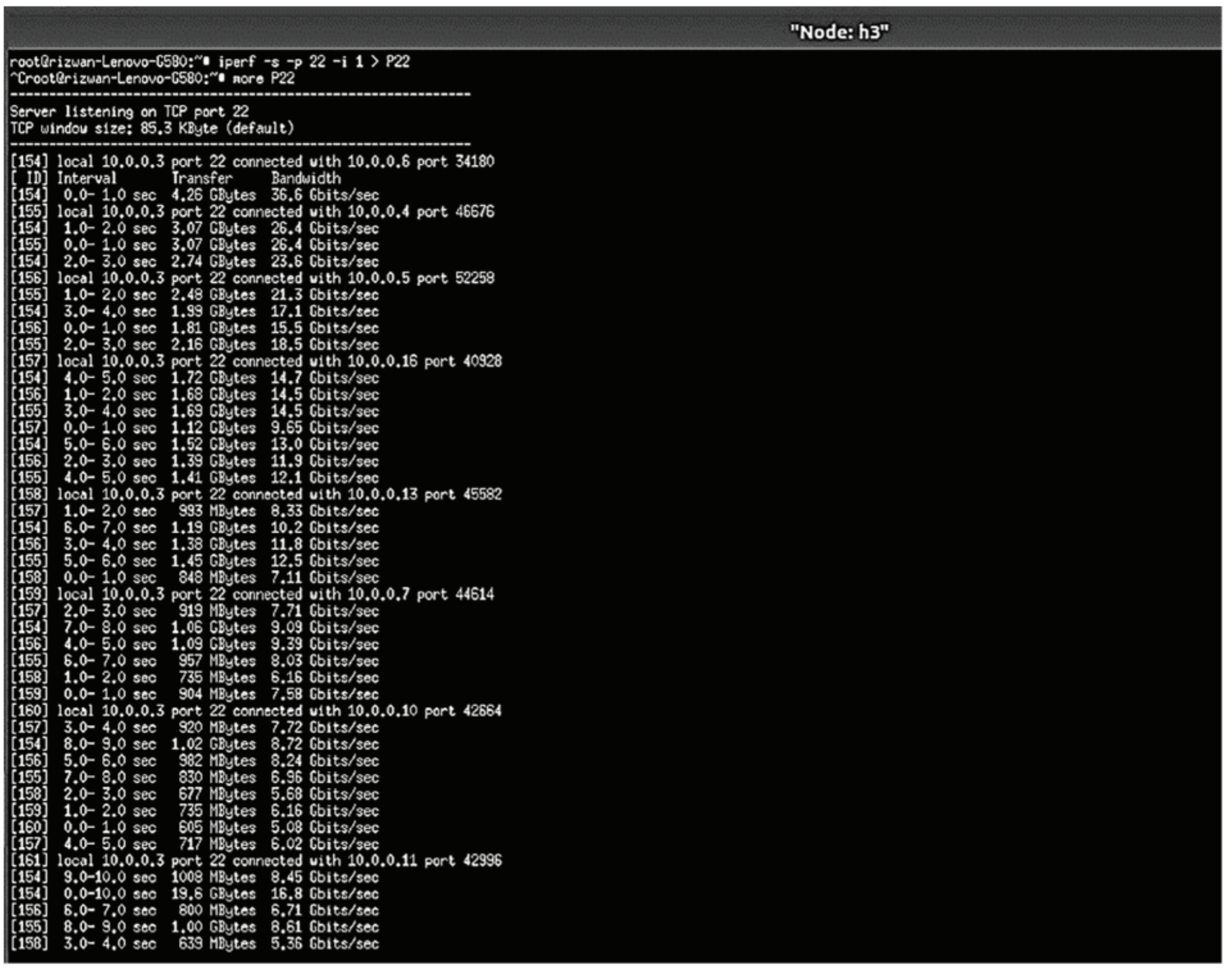


Figure 6b: Different hosts try to access port number 80 of h3

Similarly, Figs. 7a and 7b show that clients h4, h5, h6, h7, h10, h11, h16, and h17 access server h3 10.0.0.3 on port 22, the connection on port 22 from server connection establish. All the nodes working as clients have successfully approached port 22 of the h3 for 10 s and transferred data because port 22 is not a blocked-in custom firewall.

Similarly, Fig. 8 shows that client h1 accessed UDP server h3 10.0.0.3 on port 22, and the connection on port 22 from the server connection establish. h1 working as a client, has successfully approached port 22 of the h3 for 15 s and transferred data because port 22 is not a blocked-in custom firewall.

2. Using ping queries from a different host of TCP, and UDP servers, monitoring throughput, jitter, and response time. GNUPLOT is used to make a graphical representation of SDN data. Those data are retrieved by using the CAT command of Mininet.



```
"Node: h3"
root@rizwan-Lenovo-G580:~# iperf -s -p 22 -i 1 > P22
^Croot@rizwan-Lenovo-G580:~# more P22
------------------------------------------------------------
Server listening on TCP port 22
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[154] local 10.0.0.3 port 22 connected with 10.0.0.6 port 34180
[ ID] Interval       Transfer     Bandwidth
[154]  0.0- 1.0 sec  4.26 GBytes  36.6 Gbits/sec
[155] local 10.0.0.3 port 22 connected with 10.0.0.4 port 46676
[154]  1.0- 2.0 sec  3.07 GBytes  26.4 Gbits/sec
[155]  0.0- 1.0 sec  3.07 GBytes  26.4 Gbits/sec
[154]  2.0- 3.0 sec  2.74 GBytes  23.6 Gbits/sec
[156] local 10.0.0.3 port 22 connected with 10.0.0.5 port 52258
[155]  1.0- 2.0 sec  2.48 GBytes  21.3 Gbits/sec
[154]  3.0- 4.0 sec  1.99 GBytes  17.1 Gbits/sec
[156]  0.0- 1.0 sec  1.81 GBytes  15.5 Gbits/sec
[155]  2.0- 3.0 sec  2.16 GBytes  18.5 Gbits/sec
[157] local 10.0.0.3 port 22 connected with 10.0.0.16 port 40928
[154]  4.0- 5.0 sec  1.72 GBytes  14.7 Gbits/sec
[156]  1.0- 2.0 sec  1.68 GBytes  14.5 Gbits/sec
[155]  3.0- 4.0 sec  1.69 GBytes  14.5 Gbits/sec
[157]  0.0- 1.0 sec  1.12 GBytes  9.65 Gbits/sec
[154]  5.0- 6.0 sec  1.52 GBytes  13.0 Gbits/sec
[156]  2.0- 3.0 sec  1.39 GBytes  11.9 Gbits/sec
[155]  4.0- 5.0 sec  1.41 GBytes  12.1 Gbits/sec
[158] local 10.0.0.3 port 22 connected with 10.0.0.13 port 45582
[157]  1.0- 2.0 sec   993 MBytes  8.33 Gbits/sec
[154]  6.0- 7.0 sec  1.19 GBytes  10.2 Gbits/sec
[156]  3.0- 4.0 sec  1.38 GBytes  11.8 Gbits/sec
[155]  5.0- 6.0 sec  1.45 GBytes  12.5 Gbits/sec
[158]  0.0- 1.0 sec   848 MBytes  7.11 Gbits/sec
[159] local 10.0.0.3 port 22 connected with 10.0.0.7 port 44614
[157]  2.0- 3.0 sec   919 MBytes  7.71 Gbits/sec
[154]  7.0- 8.0 sec  1.06 GBytes  9.09 Gbits/sec
[156]  4.0- 5.0 sec  1.09 GBytes  9.39 Gbits/sec
[155]  6.0- 7.0 sec   957 MBytes  8.03 Gbits/sec
[158]  1.0- 2.0 sec   735 MBytes  6.16 Gbits/sec
[159]  0.0- 1.0 sec   904 MBytes  7.58 Gbits/sec
[160] local 10.0.0.3 port 22 connected with 10.0.0.10 port 42664
[157]  3.0- 4.0 sec   920 MBytes  7.72 Gbits/sec
[154]  8.0- 9.0 sec  1.02 GBytes  8.72 Gbits/sec
[156]  5.0- 6.0 sec   982 MBytes  8.24 Gbits/sec
[155]  7.0- 8.0 sec   830 MBytes  6.96 Gbits/sec
[158]  2.0- 3.0 sec   677 MBytes  5.68 Gbits/sec
[159]  1.0- 2.0 sec   735 MBytes  6.16 Gbits/sec
[160]  0.0- 1.0 sec   605 MBytes  5.08 Gbits/sec
[157]  4.0- 5.0 sec   717 MBytes  6.02 Gbits/sec
[161] local 10.0.0.3 port 22 connected with 10.0.0.11 port 42996
[154]  9.0-10.0 sec  1008 MBytes  8.45 Gbits/sec
[154]  0.0-10.0 sec  19.6 GBytes  16.8 Gbits/sec
[156]  6.0- 7.0 sec   800 MBytes  6.71 Gbits/sec
[155]  8.0- 9.0 sec  1.00 GBytes  8.61 Gbits/sec
[158]  3.0- 4.0 sec   639 MBytes  5.36 Gbits/sec
```

Figure 7a: h3 working as a server

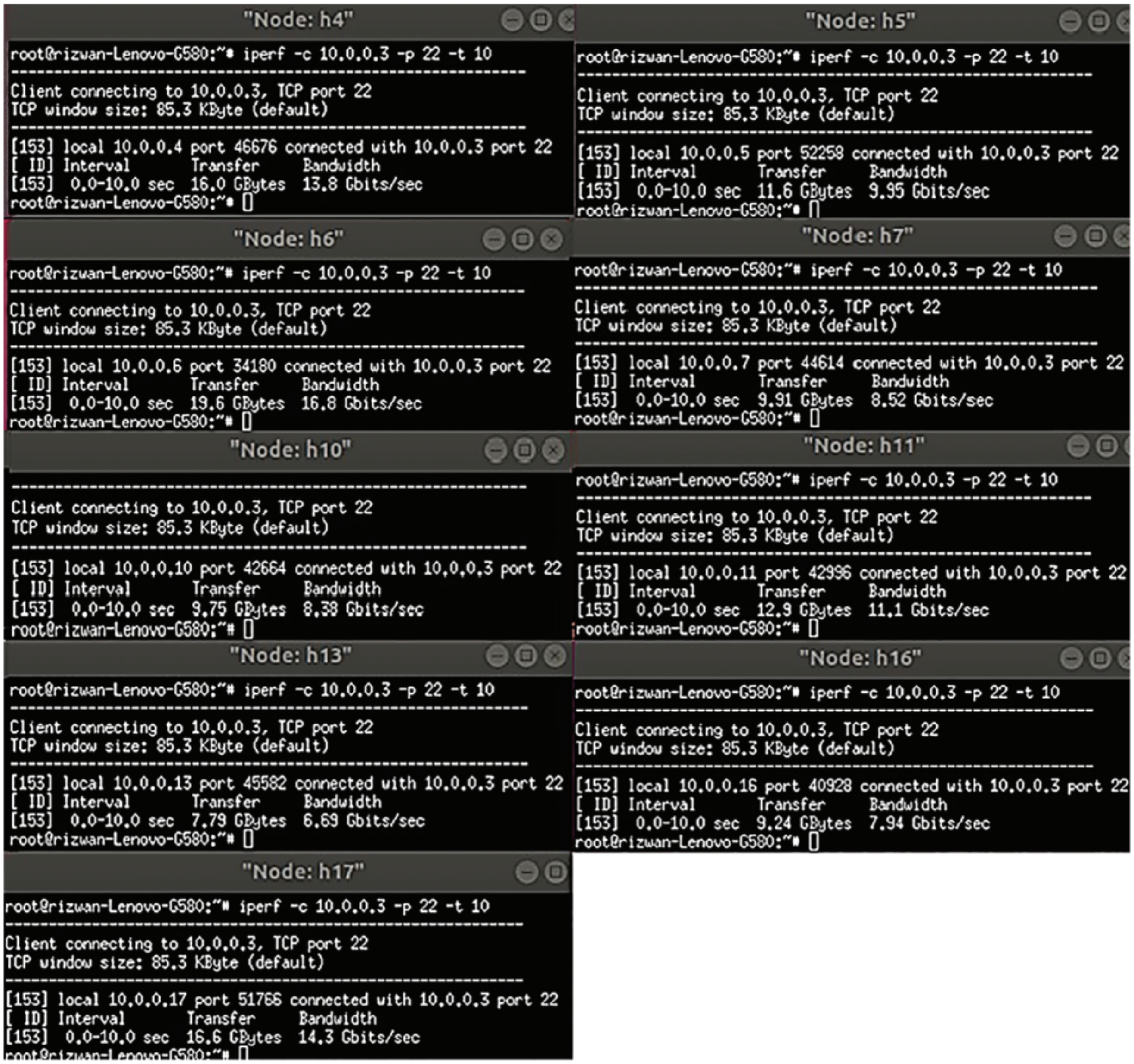


Figure 7b: Different hosts try to access port number 22 of h3 (TCP server)

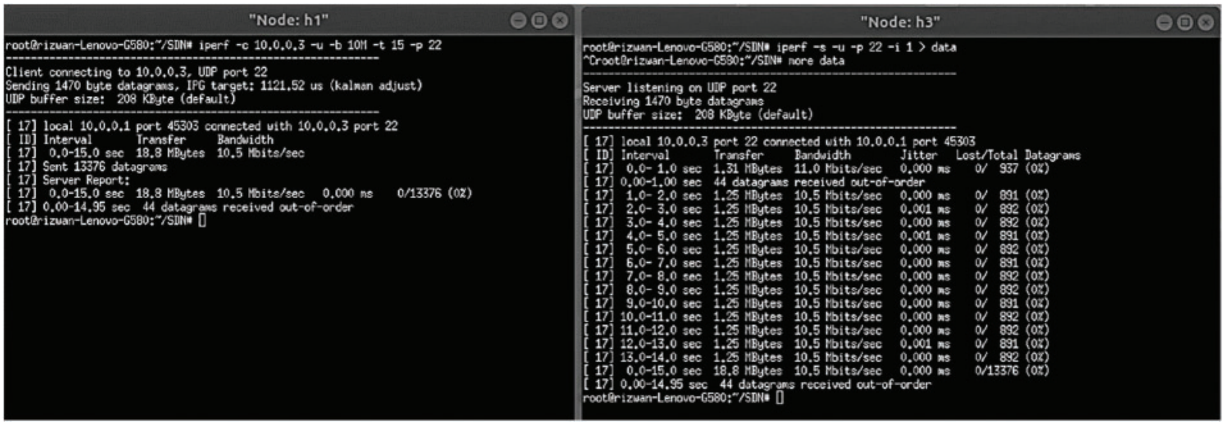


Figure 8: h3 working as a UDP server and h1 as a client

4 Evaluations

To evaluate the competence of the proposed firewall by using the ping command. Ping command measures round trip time (RTT), packet transmits, packet received, and packet loss. The comparisons have been made among the proposed firewall module and l2_learning transfer, which is a part of making OpenFlow switches act as a kind of l2_learning switches.

After the implementation of the firewall, the following parameters measure to evaluate the performance of the proposed firewall.

4.1 Round Trip Time (RTT) Evaluation

Established on layer 3 firewall rule set up on POX controller, host 3 can ping host 4. Figs. 9 and 10 show that host 3 effectively sent 10 ICMP messages to host 4 and got responses. Each output shows the same test output of POX with the firewall and without the firewall running. RTT is almost the same: 9172 ms with a firewall and 9170 ms without a firewall running. In that manner, parsing packets and matching regulations no longer influence performance.

Figure 9: ICMP traffic permitted from host 3 to host 4 with a firewall

```
root@rizwan-Lenovo-G580:~# ping -c 10 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=4.30 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.408 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.076 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.059 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.065 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.080 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.109 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=0.069 ms

--- 10.0.0.4 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9170ms
rtt min/avg/max/mdev = 0.059/0.529/4.301/1.261 ms
root@rizwan-Lenovo-G580:~#
```

Figure 10: ICMP traffic permitted from host 3 to host 4 without a firewall

4.2 Bandwidth Utilization

4.2.1 TCP Flow

A GNUPLOT shows a graphical representation of bandwidth utilization to estimate the bandwidth. The values are extracted by using the iperf command. One node is used as a server, and another as a client. Generate TCP flows without a firewall at port 80, as shown in Fig. 11. Similarly, the iperf command generates TCP traffic by implementing a firewall at port 80, as shown in Fig. 12.

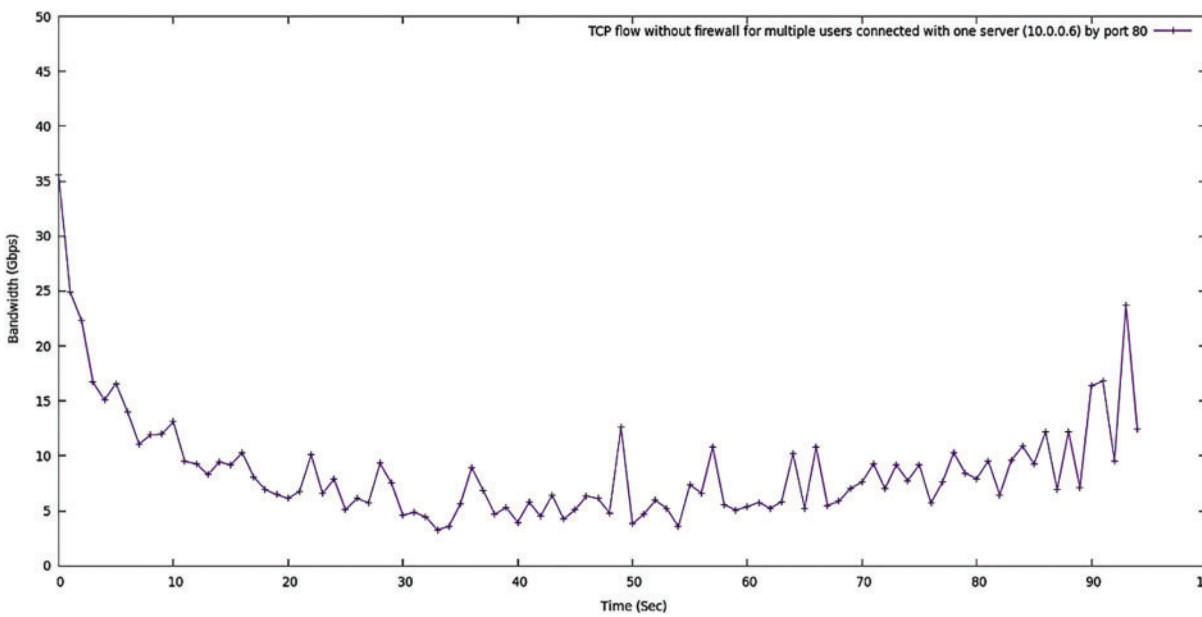


Figure 11: Bandwidth *vs.* time without a firewall (TCP flow)

As shown in Figs. 11 and 12, the data traffic appears for almost 100 s on the x-axis by accessing different clients. Consume bandwidth in Gbps is shown on the y-axis. Hence, it concludes that after implementing the proposed firewall, there is no significant consumption in bandwidth as compared to without a firewall for a TCP flow.

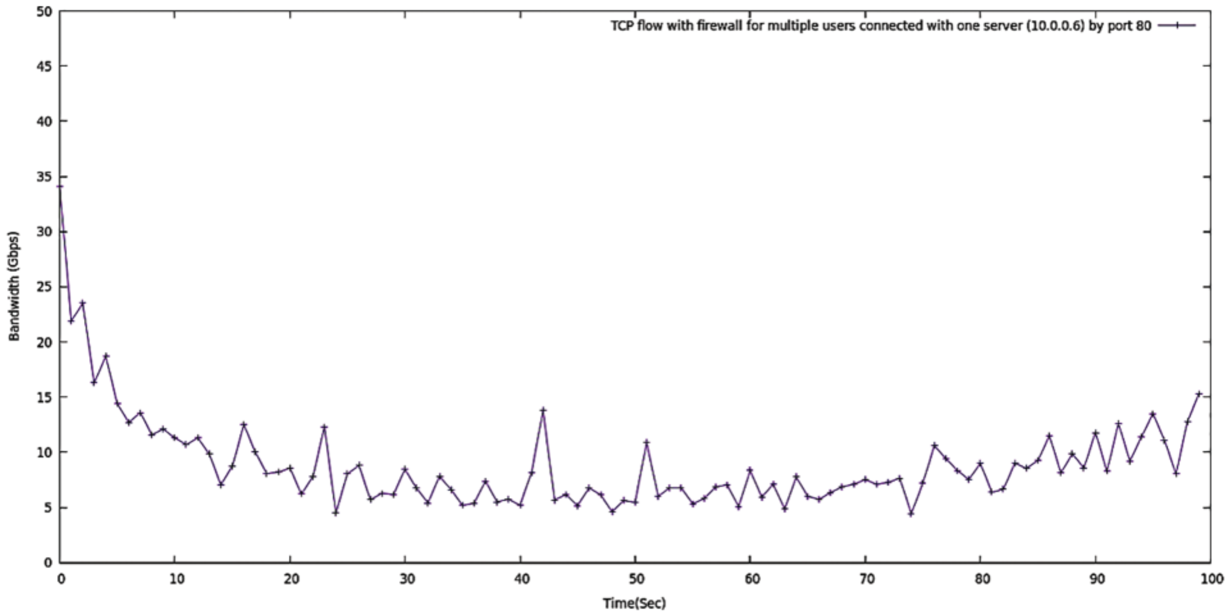


Figure 12: Bandwidth *vs.* time with a firewall (TCP flow)

4.2.2 UDP Flow

To estimate the bandwidth, GNUPLOT is used to show a graphical representation of bandwidth utilization. The values are extracted by using the iperf command. One node is used as a server, and another as a client. Generate UDP flows without a firewall at port 80, as shown in Fig. 13. Similarly, the iperf command generates UDP traffic by implementing a firewall at port 80, as shown in Fig. 14.

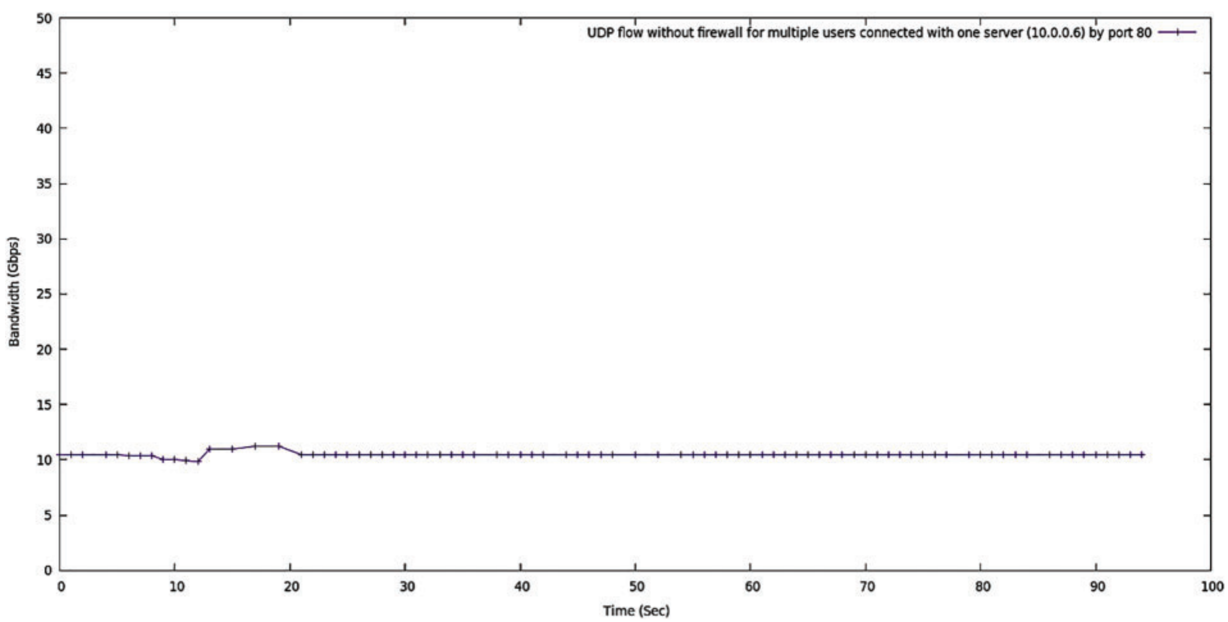


Figure 13: Bandwidth *vs.* time without a firewall (UDP flow)

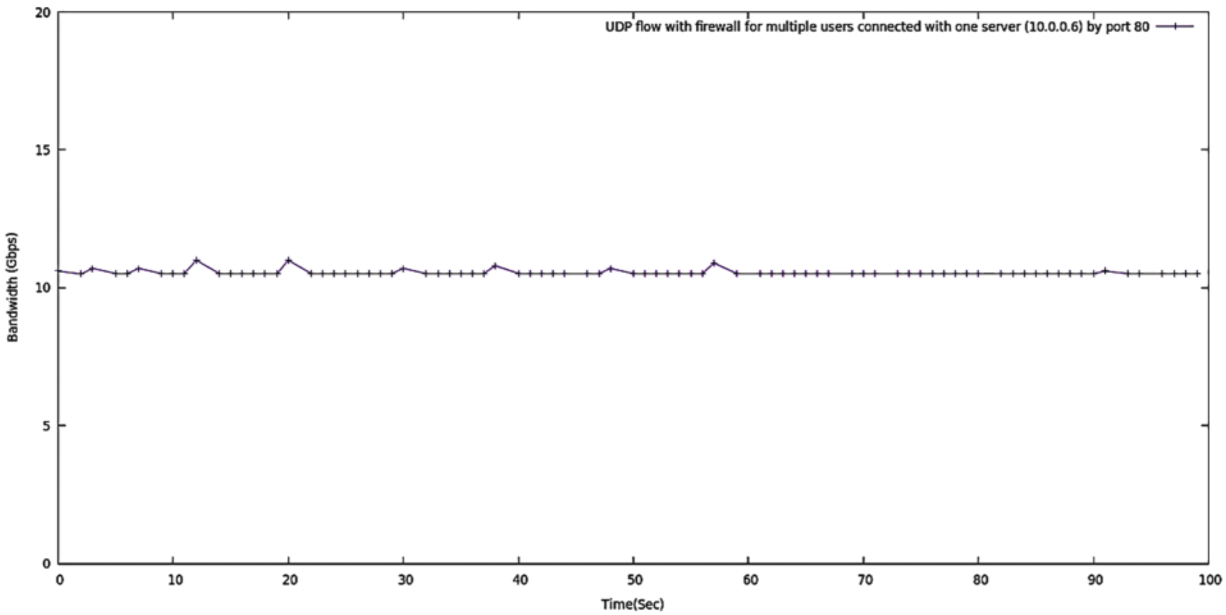


Figure 14: Bandwidth *vs.* time with a firewall (UDP flow)

As shown in Figs. 13 and 14, the data traffic appears for almost 100 s on the x-axis by accessing different clients. Consume bandwidth in Gbps is shown on the y-axis. Hence it is concluded that after implementing the proposed firewall, the graph shows a relatively small change in bandwidth consumption. Still, it shows that the proposed firewall has no adverse effect on bandwidth for UDP flow compared to the implementation of the firewall.

4.3 Latency (Jitter)

Latency or jitter is the time-dependent delay of data transmitted from sender to receiver. The values are extracted by using the iperf command. One node is used as a server, and another as a client. Generate UDP flows without a firewall at port 80, as shown in Fig. 15. Similarly, the iperf command generates UDP traffic by implementing a firewall at port 80, as shown in Fig. 16.

As shown in Figs. 15 and 16, the data traffic appears for almost 100 s on the x-axis by accessing different clients. The appearing jitter in (ms) shows on the y-axis. Hence it is concluded that after implementing the proposed firewall, the graph shows better results regarding minimum jitter values. It is a noticeable change in jitter values as compared to firewall implementations.

4.4 Data Loss

It is possible to see in Fig. 4 that the data loss is 100%. All the packets sent from h2 to h4 are unreachable. It is clear from Fig. 5 that there is a 0% packet loss from h3 to h6. All packets were received without loss. It is mentioned in firewall rules that there is no connectivity of h2 to any destination, so it shows 100% packet loss.

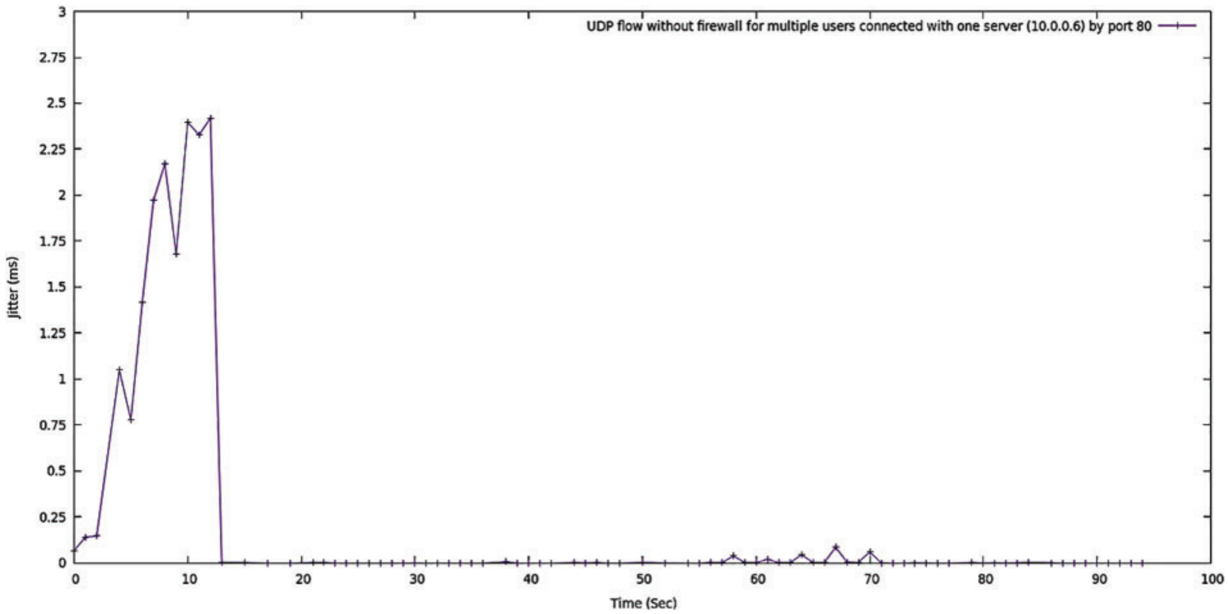


Figure 15: Jitter *vs.* time without a firewall

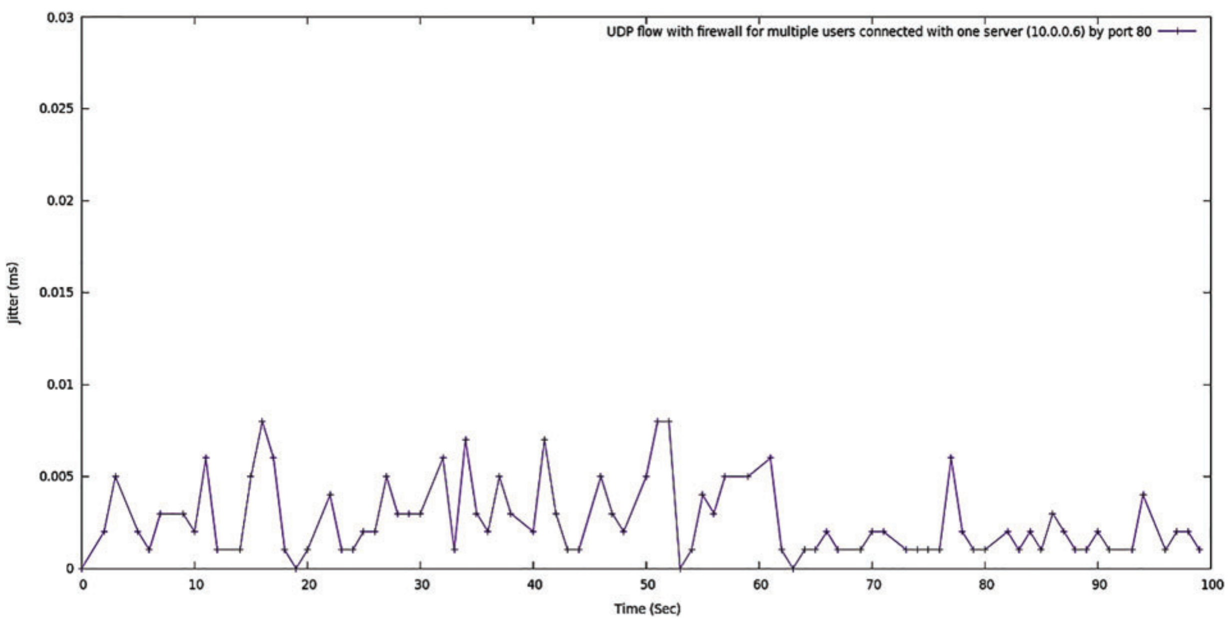


Figure 16: Jitter *vs.* time with a firewall

5 Performance Measure Metrics of SDN

Fig. 3 shows the suggested network design with 64 hosts and 10 OpenFlow switches for testing on 8 GB RAM and Linux. Ubuntu 16.04 LTS Mininet emulates the POX controller. Host-switch throughput is 100 Mbps. "iperf" and "ping" check network node connectivity and data traffic. Six infected hosts employed Low Orbit Ion Cannon (LOIC) to DDoS the network server. TCP and

UDP-based DDoS flooding attacks lasted 35 min. These attacks were tested for performance metrics. Network performance measurements forecast and prevent downtime. Table 5 shows the conventional behavior of performance parameters regarding different scenarios, i.e., normal and attack in an SDN environment.

Table 5: Conventional SDN setup

Parameters	Scenario	Behavior
Throughput	Not under attack	High
	Under attack	Low
Response time	Not under attack	Low
	Under attack	High
Jitter	Not under attack	Stable
	Under attack	Vary a lot

Table 6 shows statistics of different network parameters in normal scenarios and attack scenarios. Changes in network parameters because of attacks indicate that the SDN is evolving. Therefore, the methodology described in Section 6 uses data analysis techniques in their behavioral investigation.

Table 6: Statistics for S_{tp}, S_{rt}, and S_{jt} (over 2000 samples) for different operating scenarios

Parameters	Case of operations	Minimum	Maximum	Mean	Median	Standard deviation
Throughput	Normal TCP scenario	29.6	47.6	39.85365	39.8	2.24228691
	TCP DDoS flooding attack scenario	0	60.2	31.04907	32.1	7.701552
	Normal UDP scenario	33.7	105	104.5	105	5.721683418
	UDP DDoS flooding attack scenario	10.5	10.5	10.5	10.5	0
Round trip time	Normal TCP scenario	0.061	50.576	3.250	0.0876	12.219
	TCP DDoS flooding attack scenario	0.037	0.985	0.090	0.0953	0.021
Jitter	Normal TCP scenario	0.0020	0.2710	0.1160	0.0831	0.0043
	TCP DDoS flooding attack scenario	0.0020	0.2710	0.1160	0.0831	0.0043

(Continued)

Table 6 (continued)

Parameters	Case of operations	Minimum	Maximum	Mean	Median	Standard deviation
	Normal UDP scenario	0	4.902	0.012	0	0.201935711
	UDP DDoS flooding attack scenario	0	0.015	0.001728228	0.001	0.001897661

6 Analysis & Discussion

This section uses univariate, multivariate, and graphical methods to display and analyze the network parameters mentioned in Section 5. Data Science is the primary technique for data analysis. Using data analysis techniques, generate descriptive statistics and histograms for visualization and interpretation. Using a linear regression model to recognize and statistically for the pairwise relations between the Jitter (S_{jt}), the response time (S_{rt}), and throughput (S_{tp}) parameters of the emulated Scenario of SDN. For operating normally histogram shows a "Normal TCP Scenario", "TCP attack scenario", and "UDP Attack Scenario".

Throughput has shown in Fig. 17 for "Normal TCP Scenario", "TCP Attack Scenario", "Normal UDP Scenario", and "UDP Attack Scenario". It impacts SDN DDoS flooding attacks. When the SDN was under a TCP attack, most metrics were spread around 0, compared to a range of 5 to 55 when it usually worked or under a UDP flooding attack. S_{tp} distribution did not alter under the SDN-exposed UDP attack scenario. S_{tp} is more vulnerable to TCP and UDP attacks since they flood the target server with repeated connection requests, using its network resources and DDoS valid requests. The UDP attack scenario, in which the targeted server checks and responds to every UDP packet, including spoofed ones, may not affect S_{tp}.

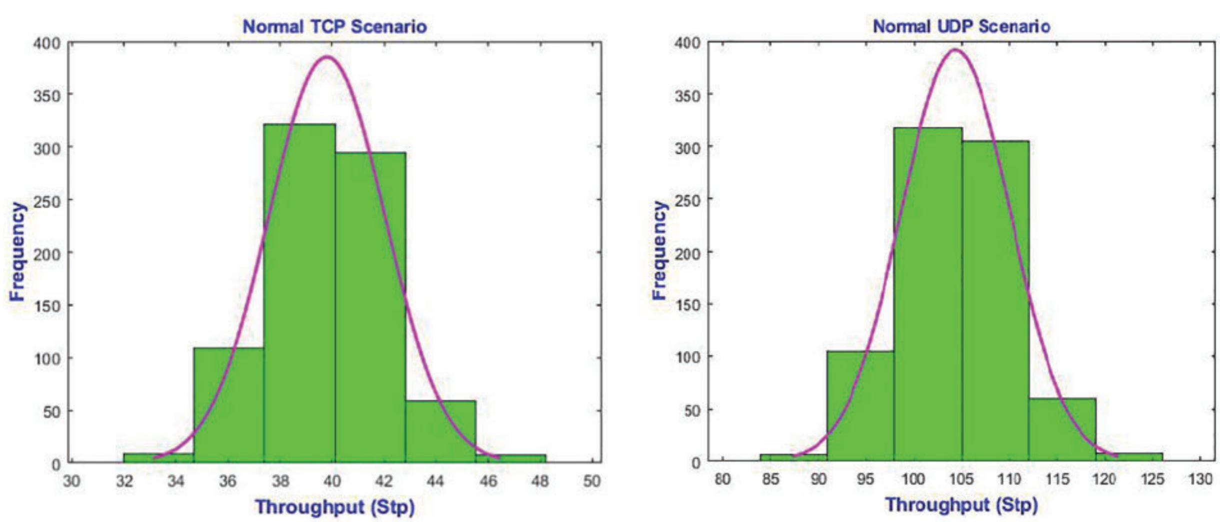


Figure 17: (Continued)

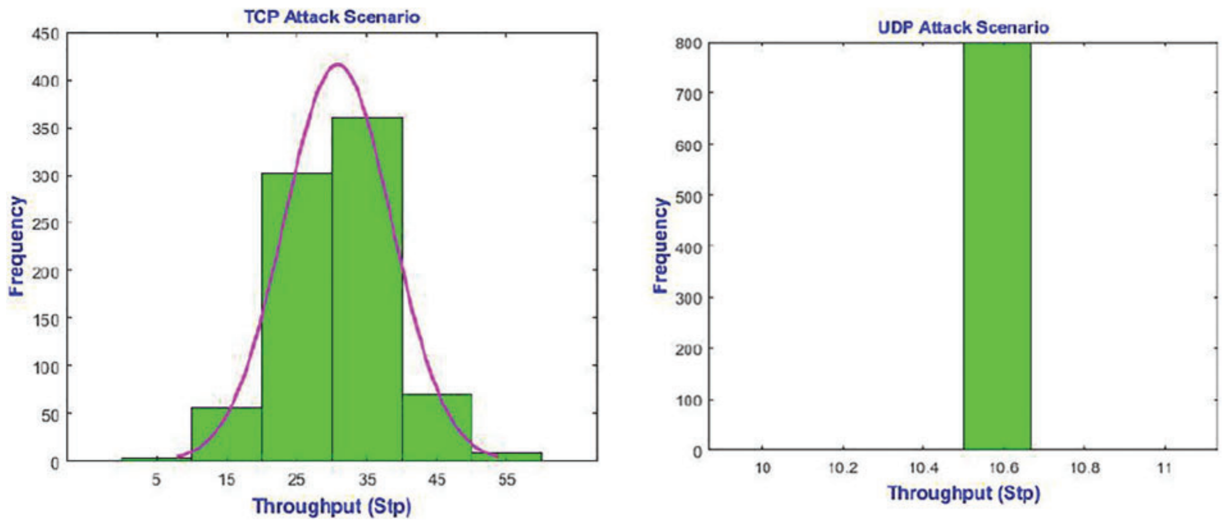


Figure 17: S_{tp} metrics for various SDN scenarios

Fig. 18 shows jitter implications. It controls the “Normal TCP Scenario”, “TCP Attack Scenario”, “Normal UDP Scenario”, and “UDP Attack Scenario”. SDN TCP flooding did not affect jitter distribution. Instead of −0.6 to 0.6, most measurements are −4 to 8 in the “Normal UDP Scenario” and “UDP Attack Scenario”. Jitter is about timing and packet sequence in a typical SDN or a TCP flooding attack. The jitter will be considerable if packets arrive in intermittent or out-of-order clusters. UDP DDoS flooding attacks are more effective against jitter.

Fig. 19 shows response time conclusions. It controls the “Normal TCP Scenario”, “TCP Attack Scenario”, and altered response time distribution. TCP flooding attacks 0.02 to 0.16 against −40 to 40 in regular SDN operation. Hence, response time is a crucial network monitoring parameter, and DDoS flooding attacks can severely impact it, notably when packets must be acknowledged before being sent again.

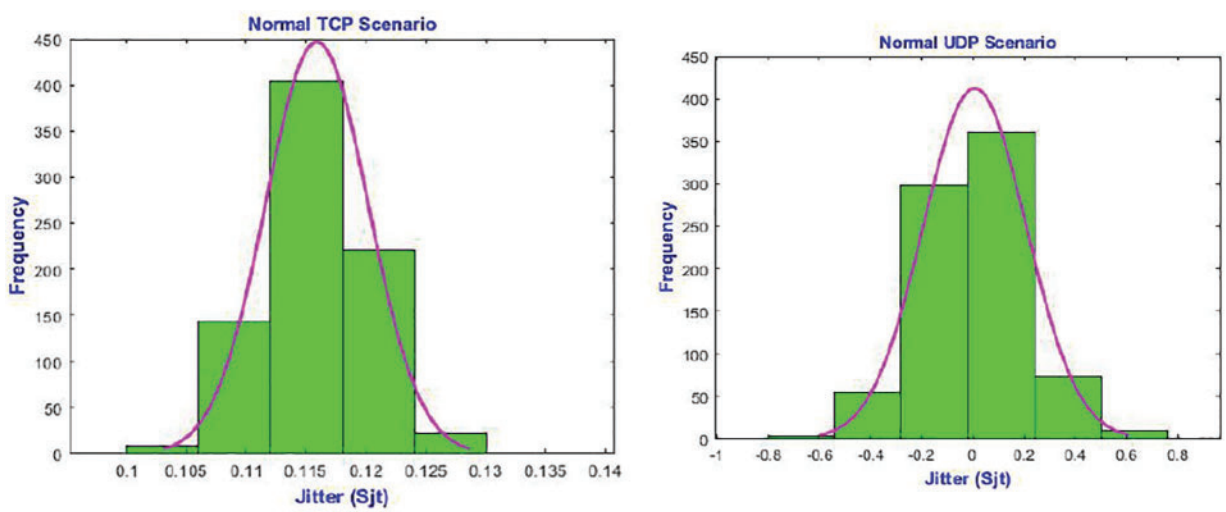


Figure 18: (Continued)

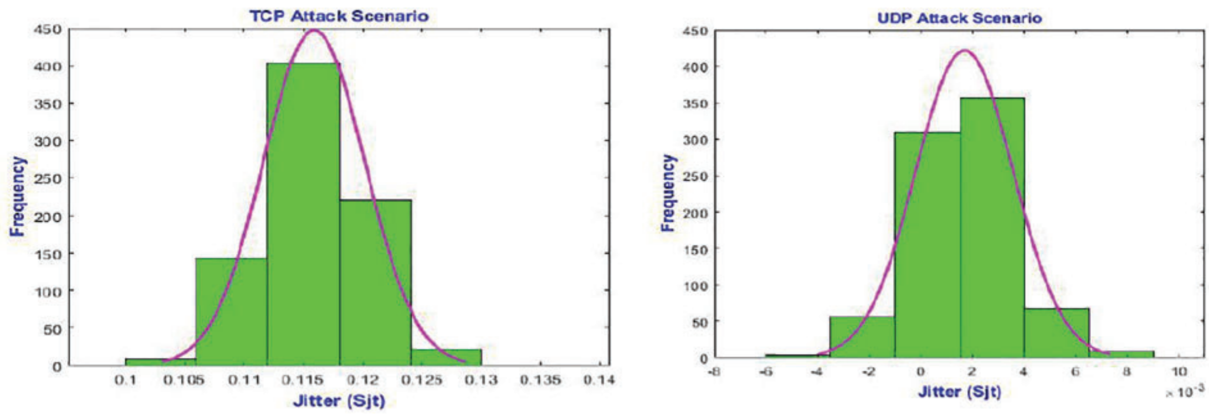


Figure 18: S_{jt} metrics for various SDN scenarios

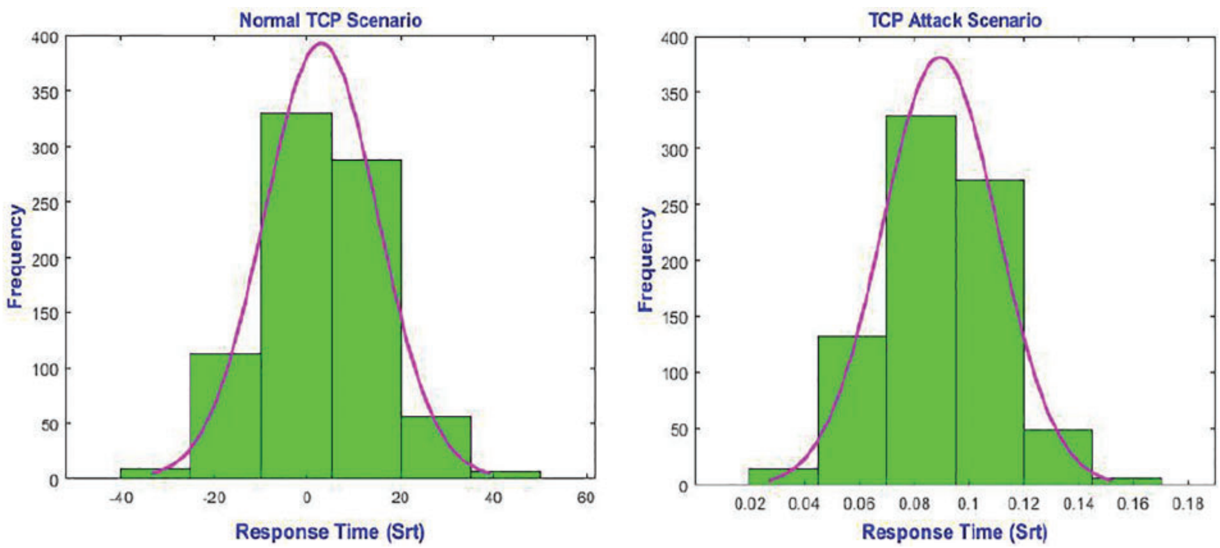


Figure 19: S_{rt} metrics for various SDN scenarios

These recommendations may be incomplete, but it is anticipated that they will be enough to educate. As research and visualization of the indicators to show the performance of SDN, administrators, and operators are informed about similar attacks on the SDN. As discussed above, it can be the most recent studies extracted from the literature to provide methods for advancing the quality of service, identifying and categorizing the status of the SDN in the face of an attack like a DDoS flooding attack. Apply Regression-based sensitivity analysis in this work. The following sections are based on a more thorough inferential assessment of the network, a description to secure the network when anomalous state changes or transitions because of attacks like DDoS flooding.

7 Conclusion

This study analyzes the performance evaluation of firewalls by applying it to different network layers. Creating topology through the POX controller using four nodes and one OpenFlow switch on a Mininet simulation tool is highly significant for SDN research. We extract results in graphical form by using a GNUPLOT. After extensive simulation study has evaluated the bandwidth of TCP flow with and without firewall implementation, the bandwidth of UDP flows with and without firewall implementation, latency, roundtrip time (RTT), and packet loss parameters. Work demonstrates that the proposed firewall significantly influences bandwidth, roundtrip time, jitter, and packet loss. The findings give SDNs operator inference-based evaluation guidelines. Even if these rules are not exhaustive, they anticipate adequate notification of SDNs operators regarding the possibility of an SDN attack based on the study and analysis of SDN performance indicators. In the future, different other controllers like RYU, Floodlight, and NOX will be used. Also, generate application layer firewall rules. Real-world, real-time SDN data will validate the investigations and findings presented in this work and serve as the foundation for creating comprehensive guidelines for SDN administrators and operators.

Acknowledgement: This work is an expanded version of "Towards Secure Implementations of SDN-Based Firewall," which appeared in the Journal of Independent Studies and Research Computing, 20(2), 40–47, 2022. This research supported by Akinsolu, M. O., Sangodoyin, A. O., and Uyoata, U. E., "Behavioral Investigation of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis," Mathematics, 10(14), 2536. This research is part of the author's Ph.D. dissertation at Hamdard University. The results were implemented on the MININET emulator from the GitHub Community Forum and online-accessible functions and procedures for the initial implementation.

Funding Statement: This research is supported in part by the Research Committee of Hamdard University Karachi Pakistan (www.hamdard.edu.pk) and the Office of Research Innovation & Commercialization (ORIC) of Dawood University of Engineering & Technology Karachi Pakistan (www.duet.edu.pk).

Author Contributions: The author's contribution to the paper is as follows: study conception and design: R. Iqbal and R. Hussain; data collection: R. Iqbal and R. Hussain; analysis and interpretation of results: R. Iqbal, S. Arif and N. M. Ansari; draft manuscript preparation: R. Iqbal, N. M. Ansari and T. A. Shaikh. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: This research is part of the author's Ph.D. dissertation at Hamdard University. Due to the nature of this research, participants of this study did not agree for their data to be shared publicly and only available upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Urrea and D. Benítez, "Software-defined networking solutions, architecture, and controllers for the Industrial Internet of Things: A review," *Sensors*, vol. 21, no. 19, pp. 1–20, 2021.

[2] M. Du and K. Wang, "An SDN-enabled pseudo-honeypot strategy for distributed denial of service attacks in the industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 648–657, 2020.

[3] A. Rahman, M. J. Islam, S. S. Band, G. Muhammad, K. Hasan *et al.,* "Towards a blockchain-SDN-based secure architecture for cloud computing in smart industrial IoT," *Digital Communications and Networks*, vol. 9, no. 2, pp. 411–421, 2022.

[4] M. Alabbad and R. Khedri, "Configuration and governance of dynamic secure SDN," *Procedia Computer Science*, vol. 184, pp. 131–139, 2021.
[5] R. Iqbal, R. Hussain and S. Arif, "Investigating the attacks on software defined networks: Summary & recommendations," *Journal of Xi'an Shiyou University, Natural Science Edition*, vol. 18, no. 9, pp. 319–326, 2022.
[6] G. Rezaei and M. R. Hashemi, "An SDN-based firewall for networks with varying security requirements," in *Proc. of IEEE CSICC*, Tehran, Iran, pp. 1–7, 2021.
[7] B. V. Baiju, S. M. Yahiya, P. A. Raj and S. S. Farooq, "DDoS attack detection using SDN techniques," *Turkish Journal of Computer and Mathematics*, vol. 12, no. 10, pp. 326–335, 2021.
[8] M. T. Islam, N. Islam and M. A. Refat, "Node to node performance evaluation through RYU SDN controller," *Wireless Personal Communications*, vol. 112, no. 1, pp. 555–570, 2020.
[9] A. Derhab, M. Guerroumi, M. Belaoued and O. Cheikhrouhou, "BMC-SDN: Blockchain-based multicontroller architecture for secure software-defined networks," *Wireless Communication and Mobile Computing*, vol. 2021, pp. 1–12, 2021.
[10] K. Siddhesh, V. Susmita and D. Pradhan, "Portable firewall for data security toward secured communication," *East African Scholars Journal of Engineering and Computer Sciences*, vol. 4, no. 4, pp. 41–45, 2021.
[11] S. Constantinou, A. Vasileiou, A. Konstantinidis, P. K. Chrysanthis and D. Z. Yazti, "IMCF: The IoT meta-control firewall for smart buildings," in *Proc. of EDBT*, Nicosia, Cyprus, pp. 658–661, 2021.
[12] S. Hafizah, S. Ariffin, N. Muazzah, A. Latiff and M. Hamed, "The impact of firewall on TCP and UDP throughput in an OpenFlow software-defined network," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 1, pp. 256–263, 2020.
[13] R. Banu, T. Jyothi, M. Amulya, K. N. Anju, A. Raju *et al.,* "MONOSEK–A network packet processing system for analysis detection of TCP xmas attack using pattern analysis," in *Proc. of ICCS*, Madurai, India, pp. 952–956, 2019.
[14] H. M. Noman and M. N. Jasim, "POX controller and open flow performance evaluation in software defined networks (SDN) using mininet emulator," in *Proc. of ICSET*, Baghdad, Iraq, pp. 1–9, 2020.
[15] R. Alkanhel, A. Ali, F. Jamil, M. Nawaz, F. Mehmood *et al.,* "Intelligent transmission control for efficient operations in SDN," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 2807–2825, 2022.
[16] M. Maray, H. M. Alshahrani, K. A. Alissa, N. Alotaibi, A. Gaddah *et al.,* "Optimal deep learning-driven intrusion detection in SDN-enabled IoT environment," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 6587–6604, 2023.
[17] H. A. Alamri, V. Thayananthan and J. Yazdani, "Machine learning for securing SDN based 5G network," *International Journal of Computer Applications*, vol. 174, no. 14, pp. 9–16, 2021.
[18] S. Ali, M. K. Alvi, S. Faizullah, M. A. Khan, A. Alshanqiti *et al.,* "Detecting DDoS attack on SDN due to vulnerabilities in OpenFlow," in *Proc. of AECT*, Al Madinah Al Munawwarah, Saudi Arabia, pp. 1–6, 2020.
[19] S. Haas, F. Wilkens and M. Fischer, "Scan correlation-revealing distributed scan campaigns," in *Proc. of NOMS*, Budapest, Hungary, pp. 1–6, 2020.
[20] Y. Li, X. Guo, X. Pang, B. Peng, X. Li *et al.,* "Performance analysis of floodlight and RYU SDN controllers under mininet simulator," in *Proc. of ICCC*, Chongqing, China, pp. 85–90, 2020.
[21] M. O. Akinsolu, A. O. Sangodoyin and U. E. Uyoata, "Behavioral study of software-defined network parameters using exploratory data analysis and regression-based sensitivity analysis," *Mathematics*, vol. 10, no. 14, pp. 1–26, 2022.
[22] A. Mishra, N. Gupta and B. B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," *Telecommunication System*, vol. 77, no. 1, pp. 47–62, 2021.
[23] K. Bhushan and B. B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," *Journal of Ambient Intelligence and Humaniz Computing*, vol. 10, no. 5, pp. 1985–1997, 2019.

[24] A. Mishra, B. B. Gupta, D. Perakovic, S. Yamaguchi and C. H. Hsu, “Entropy based defensive mechanism against DDoS attack in SDN-cloud enabled online social networks,” in *Proc. of ICCE*, Las Vegas, NV, USA, pp. 1–6, 2021.
[25] M. D. Hatagundi and H. V. Kumaraswamy, “A comprehensive survey on different attacks on SDN and mitigation approaches,” in *Proc. of ICCMC*, Erode, India, pp. 624–627, 2019.
[26] Y. Zhang, M. Cui, M. Abadeer and S. Gorlatch, “A QoS-aware routing mechanism for SDN-based integrated networks,” in *Proc. of ICOIN*, Bangkok, Thailand, pp. 287–292, 2023.
[27] C. Smera and J. Sandeep, “Networks simulation: Research-based implementation using tools and approaches,” in *Proc. of GCAT*, Bangalore, India, pp. 1–7, 2022.
[28] E. Anthi, L. Williams, M. Słowi, G. Theodorakopoulos and P. Burnap, “A supervised intrusion detection system for smart home IoT devices,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.
[29] S. Macwan and C. H. Lung, “Investigation of moving target defence technique to prevent poisoning attacks in SDN,” in *Proc. of SERVICES*, Milan, Italy, pp. 178–183, 2019.
[30] R. Iqbal, R. Hussain, S. Arif, A. A. Siddiqui and S. Akhtar, “Towards secure implementations of SDN based firewall,” *Journal of Independent Studies and Research Computing*, vol. 20, no. 2, pp. 40–47, 2022.
[31] F. N. Nife and Z. Kotulski, “Application-aware firewall mechanism for software defined networks,” *Journal of Network and Systems Management*, vol. 28, no. 3, pp. 605–626, 2020.
[32] J. Li, J. Wu, H. Jiang, W. Du and W. Jiang, “SDN-based stateful firewall for cloud,” in *Proc. of IDS*, Baltimore, MD, USA, pp. 157–161, 2020.
[33] G. Kumar and H. Alqahtani, “Machine learning techniques for intrusion detection systems in SDN-recent advances, challenges and future directions,” *Computer Modeling in Engineering & Sciences*, vol. 134, no. 1, pp. 89–119, 2023.
[34] S. M. Parveen, “Intrusion detection system in software defined networks using machine learning approach,” *International Journal of Advanced Engineering Research and Science*, vol. 8, no. 4, pp. 135–142, 2022.
[35] L. Yan, M. Ma, D. Li, X. Huang, Y. Ma *et al.,* “Certrust: An SDN-based framework for the trust of certificates against crossfire attacks in IoT scenarios,” *Computer Modeling in Engineering & Sciences*, vol. 134, no. 3, pp. 2137–2162, 2023.