The impact of firewall on TCP and UDP throughput in an openflow software defined network

H.H. Khairi, Sharifah H. S. Ariffin, N. M. Abdul Latiff, Kamaludin Mohamad Yusof, M. K. Hassan, Mohammad Rava
School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Malaysia

Article Info

Article history:

Received Jan 10, 2020
Revised Mar 15, 2020
Accepted Apr 11, 2020

Keywords:

Firewall
Forwarding policies
Software define network
TCP and UDP
Throughput

ABSTRACT

Software Defined Networking (SDN) is an emerging networking paradigm that provides more flexibility and adaptability in terms of network definition and control. However, SDN is a logically centralized technology. Therefor the control plane (i.e. controller) scalability in SDN in particular, is also one of the problems that needs further focus. OpenFlow is one of the protocol standards in SDN, which allow the separation of the controller from the forwarding plane. The control plane has an SDN embedded firewall and is able to enforce and monitor the network activity. This firewall can be used to control the throughput. However, it may affect SDN performance. In this paper, throughput will be used as a performance metric to evaluate and assess the firewall impact on two protocols; Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) that passes through the forwarding planes. The evaluations have been verified through simulating the SDN OpenFlow network using MININET. The results show that an implementation of firewall module in SDN creates a significant 36% average drop for TCP and 87% average drop for UDP in the bandwidth which eventually affect the quality of the network and applications.

Corresponding Author:

Mutaz Hamed Hussien Khairi,
Faculty of Engineering,
School of Electrical Engineering,
Universiti Teknologi Malaysia, Johor, Malaysia.
Email: hhkmutaz2@graduate.utm.my

1. INTRODUCTION

Software-defined networking (SDN) is a network management technique that enhanced the performance of the network by making its structure and configuration more dynamically and programmatically efficient [1]. The application of this approach leads to several benefits, such as dealing with the changing business requirements by allowing the administrators and network engineers to make these changes via a centralized control console [2]. SDN's implementation on a network allows it to become more flexible and agile by combining a multitude of network technologies, specifically designed for such purpose. SDN's main structure involves separating the network control from forwarding planes, which put into context would be same as separating the brain from the muscle [3]. This separation would theoretically enable the network's control plane (or the brain) to be programmable on its own, and thus provide the network engineers a direct control over the underlying infrastructure of the network [4]. SDN also has other underlying benefits, such as being manageable, dynamic, adaptable and more importantly cost effective, which make it an ideal solution for the ever-growing nature of the internet and its application's high-bandwidth requirements [5].

SDN has several standards, one of which is OpenFlow (OF). In the SDN context, one of OpenFlow's key components is the controller, which by representing the network Operating System, allows for application development through north bound API. SDN is heavily influenced by the behavior of the controller, and its efficiency is directly correlated with the efficiency of the SDN [6]. The architecture of the OpenFlow in SDN is illustrated in Figure 1. The OpenFlow switch contain a number of flow tables and are connected to the controller via an OpenFlow protocol. This protocol is used as a form of communication between the switches and the controller, which uses the flow tables as an abstraction. The purpose of the flow tables is to ensure that the packets are allocated, processed and delivered correctly by adhering to the flow entries in the flow tables [7, 8].

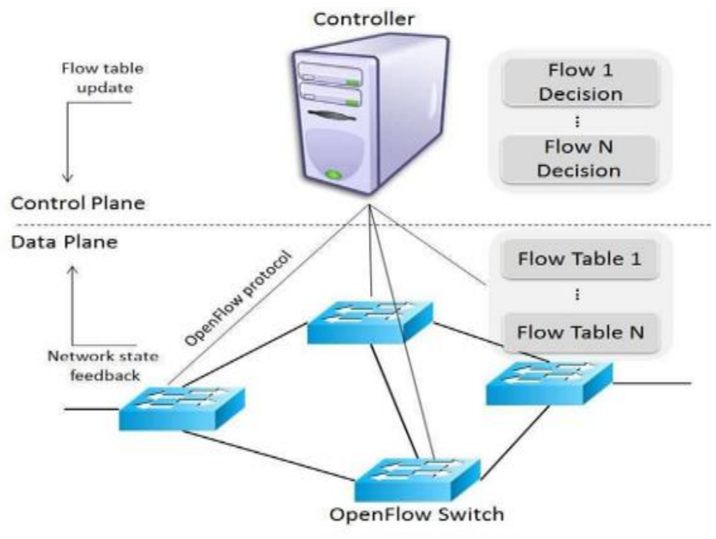

Figure 1. OpenFlow SDN architecture

Hardware Security Modules (HSM) have several functions and aspects which is to control the flow of the information and packets and communicate methods [9]. The traditional firewalls and SDN based firewalls have several differences. The SDN-based firewall filters the packets as it passes through the controller, and the packets that are subsequently followed are adhered to the controller firewall policy, which is enforced by the controller itself. SDN based firewall works both as a packet filter and a policy checker. Despite the flexibility that SDN architecture provides in terms of flexibility and programmability, SDN network with firewalls comes with penalties of extra overheads that influence the entire networks performance in terms of Quality-of-Service (QOS) such as jitter, latency and throughput [10].

The literature has studied on SDN network performances using different controller software and using different performance metric. Table 1, shows difference research in firewall for SDN. However, after completed literature review and related work for firewall in SDN, the research come out with that there are few numbers of studies that focus on the performance evaluation of embedded SDN firewall which may add extra overheads in term of latency, throughput and processing time.

Table 1, Comparison and summary of firewall related work in SDN

No	Title	Controller	Application or Metrics Performance
1	Firewall and load balancing as an application of SDN[11].	Floodlight	Load balancing
2	Intrusion detection system based on Software Defined Network firewall[12].	Floodlight	Detection of instruction
3	Development and research of the PreFirewall network application for floodlight SDN controller[13].	Floodlight	Security
4	Evaluation of firewall and load balance in fat-tree topology based on floodlight controller[14].	Floodlight	Security and Load balancing
5	Implementing a firewall functionality for mesh networks using SDN controller[15].	POX	Packets filtering
6	Comparative Analysis of UDP Traffic with and Without SDN-Based Firewall[16].	POX	Throughput
7	Implementation and performance analysis of SDN firewall on POX controller[17].	POX	Throughput
8	Building firewall over the software-defined network controller[18].	POX	Action on packet header

In network traffic the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) used to transmit the packets between different devices in the network. The (TCP) is a bidirectional connection which the source host will receive acknowledgement from the destination host after the complete packets are received. The performance of TCP and UDP protocol are measured in term of bandwidth and transfer rate. The (UDP) is used by apps to transfer a faster flow of information between nodes with monitoring and checking the error rate[19]. In research [20] a simulation is performed in order to demonstrate and compare between that TCP and UDP in SDN networks. TCP and UDP flows are used to check performance of SDN by using two analytical models [21]. Firewall performance evaluated in software define network using POX controller. There are some Researches indicates that the bandwidth and transfer rate can be used to evaluate and analyze software define network for two cases with and without firewall implementation [16, 17]. However, the focus of this paper is solely on the software aspect of the network, thus, the hardware security aspect is out of the scope of this research. In overall, it was found that the effects of firewall on the performance of SDN controller is less researched.

2. PROPOSED METHOD

Measuring and evaluating the throughput of any network is important in order to define the performance of the network itself. Existing researches, use ONOS controller in order to build the network in Mininet software, in order to evaluate the TCP and UDP traffic flows in a software defined network environment, using alternative firewall settings [22]. Also in research [23] and [24] the researchers used ONOS controller in mininet software to studies the bandwidth in SDN. The throughput has been tested for linear topology network on using by used link aggregation method [25].In the research[26] TCP observed to evaluate the load balance in data center SDN. There are also further researches that evaluate SDN networks regarding their use of firewall deployment settings [14, 16, 27-30].

The main objective of this research, is to evaluate and observe the throughput parameter for TCP and UDP in SDN. The bandwidth and transfer rate are measured and analyzed based on the existence of the firewall module. The controller used is Floodlight with a tree topology network, all of which is built and implemented in the Mininet software.

The flowchart for the proposed method is illustrated in Figure 2. The proposed method flowchart starts by first creating an SDN in Mininet, once created, the controller and OpenFlow switch are defined. The next step sets the firewall state, there are generally two states (ON/OFF). If the setting is set to “on”, then the server clients is configured, then the SDN is initialized, which is followed by the packets being sent in the network. Once this process is complete, the UDP and TCP iPerf are measured and recorded. Once the data is recorded, the firewall state is changed to “off” and the process is repeated until the data is recorded again. With the data available from both firewall settings, the bandwidth and transfer rate are analyzed.

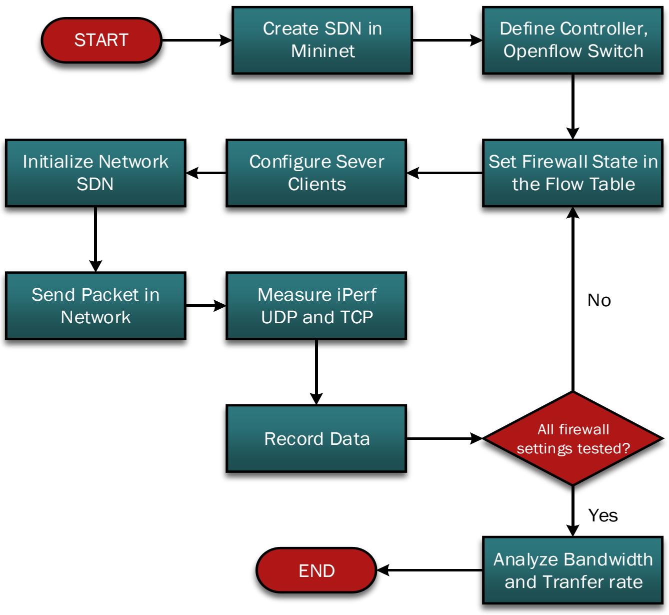
START
Create SDN in Mininet
Define Controller, Openflow Switch
Set Firewall State in the Flow Table
Configure Sever Clients
Initialize Network SDN
Send Packet in Network
Measure iPerf UDP and TCP
Record Data
All firewall settings tested?
No
Yes
Analyze Bandwidth and Tranfer rate
END

Figure 2. Proposed method flowchart

3. METHODOLOGY

In this section the System Specification and simulation design of the research are discussed as well the structure and system of the conducted experiment is also discussed in this section.

3.1. System specification

The research objective of this work is to evaluate and observe the throughput parameter in SDN with and without using the firewall module for normal forwarding SDN. The tools and applications of this work is using Mininet [31]. The Mininet simulator is used in conjunction with Floodlight [32], which is a tool used for the network controller, and Virtual Box [33], which provides a platform in Mininet virtual network can operate on. Eclipse IDE, which is a java programming tool is also use. The simulation specification environment is listed in Table 2.

Table 2. System and environment specification for MININET simulation

Software and Hardware	Specifications
Processor	Core i5
RAM	8 GB
Operation System	Ubuntu 14.04
SDN controller	Floodlight 1.2
OpenFlow Switch	Version 1.3

3.2. Simulation design

In the simulation, the forwarding SDN module has two types of policies. The first setting has firewall operating and running, while in the second setting, the firewall is disabled. In the Mininet, the SDN network is created with all of its related elements, such as the host, switches, links, controller and others. The Mininet uses python script to customize its topology, in order to customize and design network topology for various simulation scenarios. Figure 3, is the topology in Mininet software environment. This topology consists of several elements. There are five switches; *s9, s10, s11, s12, and s13*, which are connected to six hosts; *h1, h2, h3, h4, h5, and h6*. Switches *s9 and s10*, are connected to the main switch, *s1*, which is connected directly to the SDN controller, *c0*. In the simulation, two hosts, *h1 and h3*, are used as the clients that check the flow entry in OpenFlow switch in two scenarios: (a) Normal forwarding network is running. (b)The firewall module is running with policy.

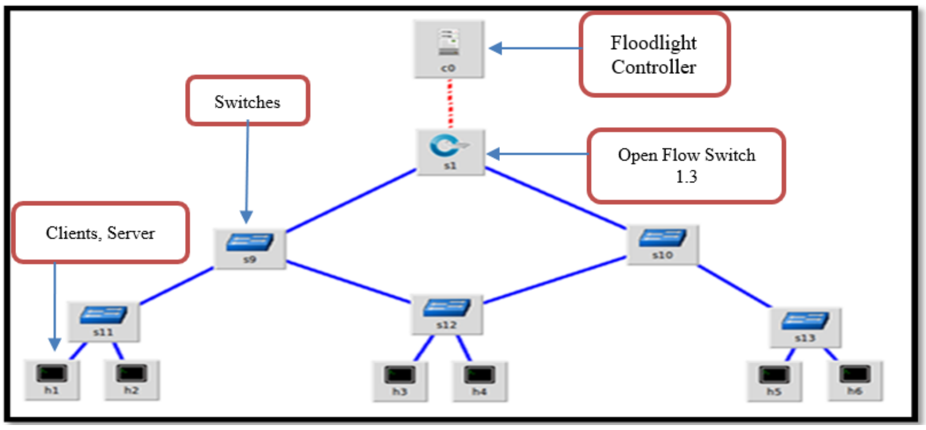

Figure 3. The network topology used in MININET

4. RESULTS AND DISCUSSION

In this section, the simulations are conducted using network topology in Figure 3 and following the simulation process flowchart depicted in Figure 2.

4.1. Results

Bandwidth and transfer rate are used as the performance measures. The simulations are done in two scenarios which are using server without and with firewall using both TCP and UDP protocols. In the first scenario, the simulation uses the IP range between 10.0.0.1 to 10.0.0.6 without firewall module. The TCP forwarding server listens on TCP port 5001, TCP window size: 85.3 KByte (default) with local 10.0.0.1 port 5001 connected with 10.0.0.3 port 36742. The second scenario of the simulation uses floodlight controller with ports 5001, 5566 in the IP range of between 10.0.0.1 to 10.0.0.6 and the firewall IP of 255.255.255.0 of class C. The following statement will enable embedded firewall functionality in the SDN

```
curlhttp://localhost:8080/wm/firewall/module/enable/json -X PUT -d ''
```

This statement located in flow table and it stop all flow entry between any switches or hosts unless it is defined as new rules which it showing in the two flowing statements:

```
curl-X POST -d '{"src-ip": "10.0.0.1/32", "dst-ip": "10.0.0.3/32"}
http://localhost:8080/wm/firewall/rules/json

curl -X POST -d '{"src-ip": "10.0.0.3/32", "dst-ip": "10.0.0.1/32"}'
http://localhost:8080/wm/firewall/rules/json
```

In this scenario the TCP traffic are redirected first to the firewall module then it will listen on TCP port 5001, TCP window size: 85.3 Kbyte (default) local 10.0.0.1 port 5001 connected with 10.0.0.3 port 44574. Figure 4 shows throughput for TCP traffic between host and the server directly in Mbits per seconds for every one interval seconds. The results are compared between TCP forwarding throughput of the server with and without firewall. Figure 5 shows the results from the same simulation but for TCP forwarding transfer.

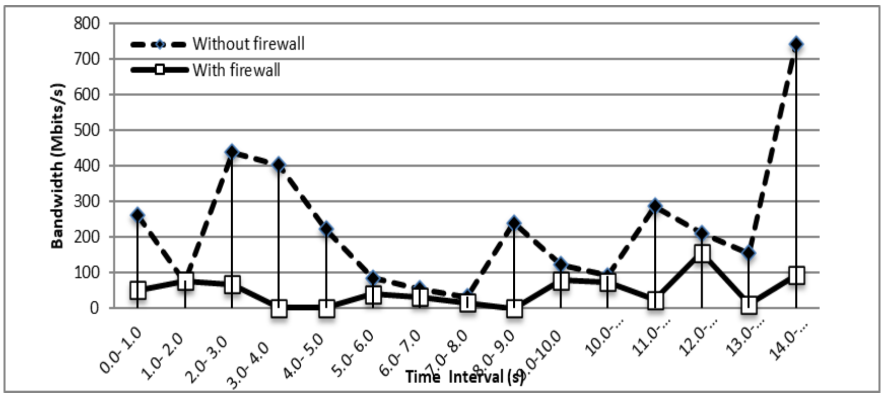

Figure 4. Comparison of TCP bandwidth forwarding throughput

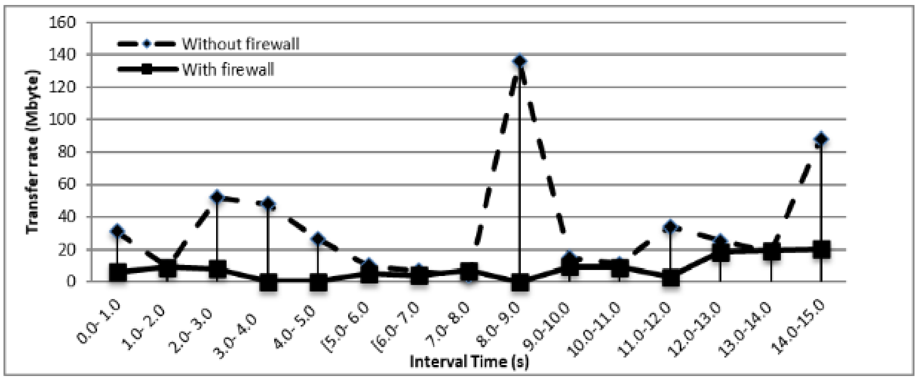

Figure 5. Comparison of TCP transfer rate forwarding throughput

In the first scenario, simulation is also implemented UDP protocol. The UDP forwarding server listens on UDP port 5566, receiving 1470-byte datagrams. The UDP buffer size by default is 208 Kbyte for local IP address 10.0.0.1. UDP port 5566 is connected with IP address 10.0.0.3 port 41408. In scenario 2, the UDP traffic are redirected to the firewall module then to the server which will be listening on UDP port 5566 and receiving 1470-byte datagram. The UDP default buffer size is 208 Kbyte, local 10.0.0.1 port 5566 is connected with 10.0.0.3 port 47282.

Figure 6 shows bandwidth throughput for UDP traffic between host and the server directly for scenario with and without firewall in Mbit per seconds for every 1 second interval time. Figure 7 presents the result comparison for UDP transfer traffic between the host and the server directly for scenario with and without firewall in Mbyte seconds for every 1 second interval time.

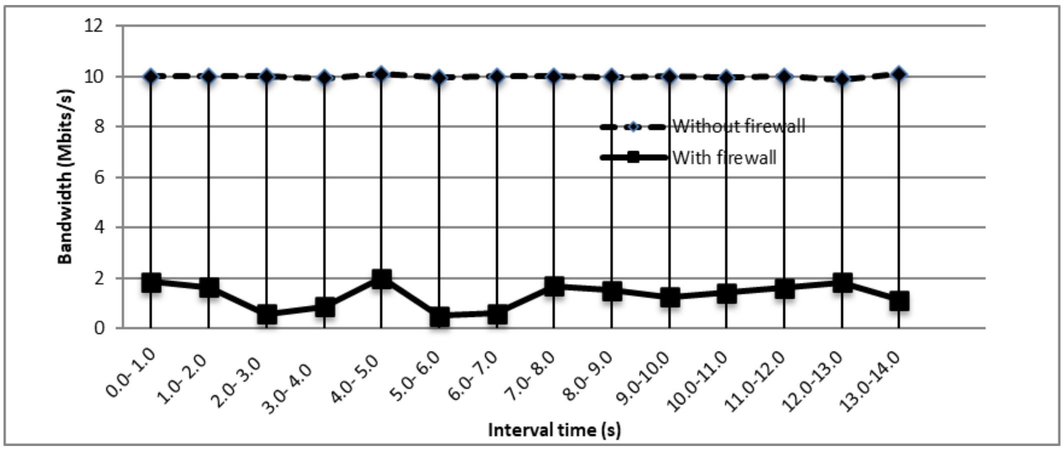

Figure 6. Comparison of UDP bandwidth forwarding throughput

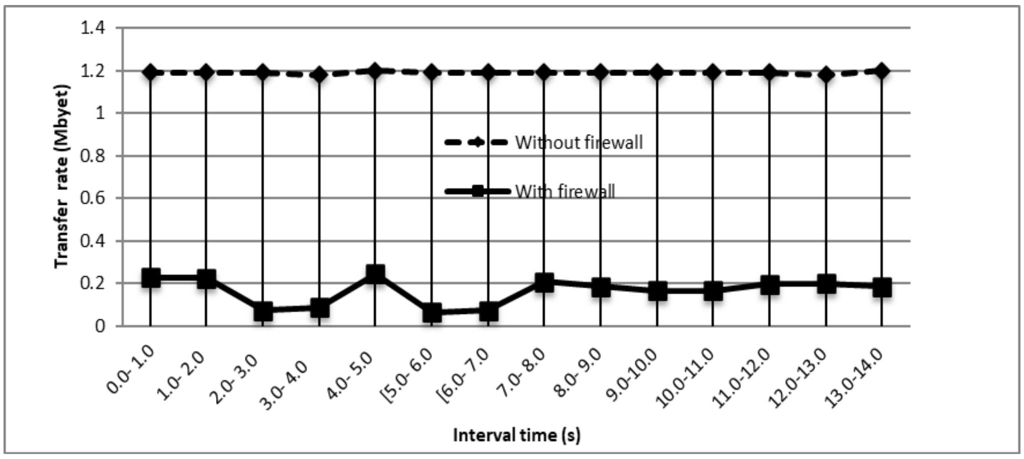

Figure 7. Comparison of UDP transfer rate forwarding throughput

4.2. Discussion

The simulation of the two scenarios using TCP and UDP protocol are measured in term of bandwidth and transfer data. Figure 4 shows the comparison of TCP forwarding throughput it shows significant of 36% average drop in throughput due to traffic redirection to the embedded firewall. This is due to the extra load in order to check the OpenFlow flows and the drop is also observed in Figure 5 for TCP transferred data 52%. Figure 6 shows the results comparing UDP protocol and there is a significant traffic drop of 87% average. This is because UDP can have significant impact on sensitive applications such as real time and interactive communications. Figure 7 shows the comparison for UDP data transfer with firewall also provide decrement in data transfer 86%. Table 3 shows the Comparison of the average Drop in TCP and UDP protocols when the two scenarios were applied.

Table 3. Comparison of the average Drop in TCP and UDP protocols with firewall

Metric	TCP	UDP
Bandwidth	36%	87%
Transfer Rate	52%	86%

In related work section, Table 2 shows that there are a few numbers of research that has been conducted in evaluation the performance of firewall in term of throughput in SDN. Two researches [16] and [17], were conducted for throughput evaluation for firewall in SDN, the experiment scenarios were based on POX controller connected with open flow switch version 1.0 in linear topology network. Therefore, this paper is to study and evaluate the effects of enabling the embedded firewall on floodlight controller connected with OpenFlow switch version 1.3 in tree topology network in terms of overall network throughput.

5. CONCLUSION

The paper presents evaluation the throughput and data transfer parameter in SDN for with and without using firewall module in normal forwarding SDN. The aim has been achieved through conducting a numerical simulation using the Mininet software simulator and the work has indicated that the implementation of a firewall has a significant impact on the bandwidth and data transfer. Observing the effect of the firewall module in Software Define Network, the benefits as well as the disadvantages of firewall in SDN could be more clearly identified. This firewall module is mainly used for purpose of security as opposed to normal forwarding network without the firewall in place.

The network simulation in Mininet shows that the throughput drop is due to firewall implementation. Although the firewall module is for security benefits, more careful considerations need to be made to avoid the adverse effects on the traffic throughput. The results indicate a clear issue that exist within firewall configuration and setting, and thus require further study that can address this issue by removing the observed delay. One of the conclusions come out of from this research that If the application or services used in SDN need to increase the transfer rate the firewall module need to consider in implementation to be not complex or lighter.

For future work, the plan is to dive deeper into the security aspects of the SDN network. There are several avenues of study that can be taken into consideration that won’t affect the QoS of the network, while at the same time can remain secure via the usage of the firewall. These changes need to occur either at the firewall module, or at the SDN controller in order to be able to provide the benefits provided by SDN, while remaining secure. Other challenges related to SDN can also be further studies and taken into consideration. Such as the main issue such like flow entries and rule conflict, which is one of the most researched topics in SDN, as it defects the security for hole SDN network and specially the rules of firewall modules.

ACKNOWLEDGEMENTS

The authors would like to thank all those had contributed towards making this research successful, especially the support from research team Computer Networks and System (CSNET). The authors wish to express their gratitude to the Ministry of Education Malaysia (MOE) and Research Management Center UTM (RMC).

REFERENCES

[1] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994-2008, 2017.
[2] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE network*, vol. 30, pp. 58-65, 2016.
[3] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Computer Networks*, vol. 112, pp. 279-293, 2017.
[4] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks: an authoritative review of network programmability technologies*, O'Reilly Media, Inc., 2013.
[5] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27-51, 2015.
[6] M. Cheng, et al., "Flow Setup Rate Test for OpenFlow Controller," 2017. [Online] Available: https://tools.ietf.org/html/draft-cheng-oftest-cont-rate-00
[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634, 2014.
[8] N. McKeown, *et al*., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.
[9] J. Attridge, "An overview of hardware security modules," *SANS Institute, InfoSec Reading Room*, vol. 1, no. 1, pp. 1-10, 2002.
[10] M. Jarschel, *et al*, "Modeling and performance evaluation of an OpenFlow architecture," *2011 23rd International Teletraffic Congress (ITC)*, pp. 1-7, 2011.

[11] N. Zope, S. Pawar, and Z. Saquib, "Firewall and load balancing as an application of SDN," *2016 Conference on advances in signal processing (CASP)*, pp. 354-359, 2016.
[12] M. A. Sayeed, M. A. Sayeed, and S. Saxena, "Intrusion detection system based on Software Defined Network firewall," *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, pp. 379-382, 2015.
[13] S. V. Morzhov and M. A. Nikitinskiy, "Development and research of the PreFirewall network application for floodlight SDN controller," *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*, pp. 1-4, 2018.
[14] S. H. Mohammed and A. D. Jasim, "Evaluation of firewall and load balance in fat-tree topology based on floodlight controller," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, pp. 1157-1164, 2020.
[15] A. Kumar and N. Srinath, "Implementing a firewall functionality for mesh networks using SDN controller," *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pp. 168-173, 2016.
[16] M. F. Monir and S. Akhter, "Comparative Analysis of UDP Traffic With and Without SDN-Based Firewall," *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp. 85-90, 2019.
[17] W. M. Othman, H. Chen, A. Al-Moalmi, and A. N. Hadi, "Implementation and performance analysis of SDN firewall on POX controller," *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 1461-1466, 2017.
[18] M. Suh, *et al*, "Building firewall over the software-defined network controller," *16th International Conference on Advanced Communication Technology*, pp. 744-748, 2014.
[19] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777-1799, 2007.
[20] M.-H. Wang, L.-W. Chen, P.-W. Chi, and C.-L. Lei, "SDUDP: A reliable UDP-Based transmission protocol over SDN," *IEEE Access*, vol. 5, pp. 5904-5916, 2017.
[21] Y.-C. Lai, *et al*, "Performance modeling and analysis of TCP and UDP flows over software defined networks," *Journal of Network and Computer Applications*, vol. 130, pp. 76-88, 2019.
[22] M. T. Naing, *et al*, "Evaluation of TCP and UDP Traffic over Software-Defined Networking," *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 7-12, 2019.
[23] M. N. M. M. N. Fathul Arif Kamarudin, Fuead Ali, "A comparative study for bandwidth on demand using ONOS Reactive and Intent forwarding," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 3, pp. 1410-1421, 2020.
[24] H. T. Zaw and A. H. Maw, "Traffic management with elephant flow detection in software defined networks (SDN)," *International Journal of Electrical & Computer Engineering*, vol. 9, no. 4, pp. 3230-321, 2019.
[25] I. D. Irawati, S. Hadiyoso, and Y. S. Hariyani, "Link Aggregation Control Protocol on Software Defined Network," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 5, pp. 2706-2712, 2017.
[26] T. E. Ali, A. H. Morad, and M. A. Abdala, "Load balance in data center SDN networks," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 5, pp. 3086-3091, 2018.
[27] R. N. Smith and S. Bhattacharya, "Firewall placement in a large network topology," *Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp. 40-45, 1997.
[28] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith, "Implementing a distributed firewall," *Proceedings of the 7th ACM conference on Computer and communications security*, pp. 190-199, 2000.
[29] L. Yuan, H. Chen, J. Mai, C.-N. Chuah, Z. Su, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pp. 15-213, 2006.
[30] S. Lee, M. Purohit, and B. Saha, "Firewall placement in cloud data centers," *Proceedings of the 4th annual Symposium on Cloud Computing*, pp. 1-2, 2013.
[31] [Online] Available: http://mininet.org/download/#option-1-mininet-vm-installation-easy-recommended.
[32] [Online] Available: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM.
[33] [Online] Available: https://www.virtualbox.org/wiki/Downloads.