Bulletin of Electrical Engineering and Informatics
Vol. 8, No. 4, December 2019, pp. 1507~1516
ISSN: 2302-9285, DOI: 10.11591/eei.v8i4.1578

On the latency and jitter evaluation of software defined networks

Paulson Eberechukwu Numan[1], Kamaludin Mohamad Yusof[2], Muhammad Nadzir Bin Marsono[3], Sharifah Kamilah Syed Yusof[4], Mohd Husaini Bin Mohd Fauzi[5], Salawu Nathaniel[6], Elizabeth N. Onwuka[7], Muhammad Ariff Bin Baharudin[8]
[1,2,3,4,5,8]Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Malaysia
[6,7]Federal University of Technology, Minna, PMB 65, Minna, Niger State, Nigeria

Article Info

Article history:

Received Mar 29, 2019
Revised Jun 26, 2019
Accepted Jul 19, 2019

Keywords:

Jitter
Latency
Mininet
Next generation networks
Software defined networking
Conventional networks

ABSTRACT

Conventional networking devices require that each is programmed with different rules to perform specific collective tasks. Next generation networks are required to be elastic, scalable and secured to connect millions of heterogeneous devices. Software defined networking (SDN) is an emerging network architecture that separates control from forwarding devices. This decoupling allows centralized network control to be done network-wide. This paper analyzes the latency and jitter of SDN against a conventional network. Through simulation, it is shown that SDN has an average three times lower jitter and latency per packet that translate to improved throughput under varying traffic conditions.

Corresponding Author:

Kamaludin Mohamad Yusof,
Advanced Telecommunication Technologies Research Group,
School of Electrical Engineering, Faculty of Engineering,
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Malaysia.
Email: kamalmy@utm.my

1. INTRODUCTION

Research on the future network is still on-going for solutions that will enable networks with better flexibility, dynamic, and support of programmable resources [1-3]. Internet-of-Things (IoT) would place tremendous challenges to the conventional network to accommodate IoT devices, which has limited battery life and limited computing capability. IoT devices such as sensor nodes with Wi-Fi network adapters, motion sensors, cameras, microphones and other types of electronic instrumentation. The network supporting IoT devices also have to accommodate IoT-specific network tasks. Hence, Software Defined Networking (SDN) [3, 4] as an emerging networking architecture that are highly scalable and flexible can be an alternative for dynamic IoT network solution [5].

The concept of SDN decouples the control plane (CP) of conventional network devices from their data plane (DP) as shown in Figure 1. The CP takes care of all routing decisions while the DP performs packet forwarding and network control based on policy determined by the CP. With this separation, the configuration devices done individually on devices has been moved towards the centralized control unit [6, 7]. OpenFlow is one such protocol that enables the implementation of the SDN [8]. OpenFlow (OF) was proposed as a clean slate protocol [9] and has been defined as the first standard by Open Networking Foundation (ONF).

This paper analyzes the latency and jitter of SDN against a conventional network for connecting IoT devices. Through simulation, it is shown that SDN has three times lower jitter and latency/packet that translate to improved throughput under varying traffic conditions. The remainder of this paper is organized as follow. The SDN paradigm, its motivation, describing the basic elements, interfaces and APIs are discussed in Section 2. An OpenFlow enabled SDN architecture for IoT is detailed out in Section 3. In Section 4, the OF based network model is proposed and later, simulation on Mininet is also discussed. In Section 5, the simulation results are analyzed and discussed. Lastly, Section 6 concludes the paper and suggests potential future research.

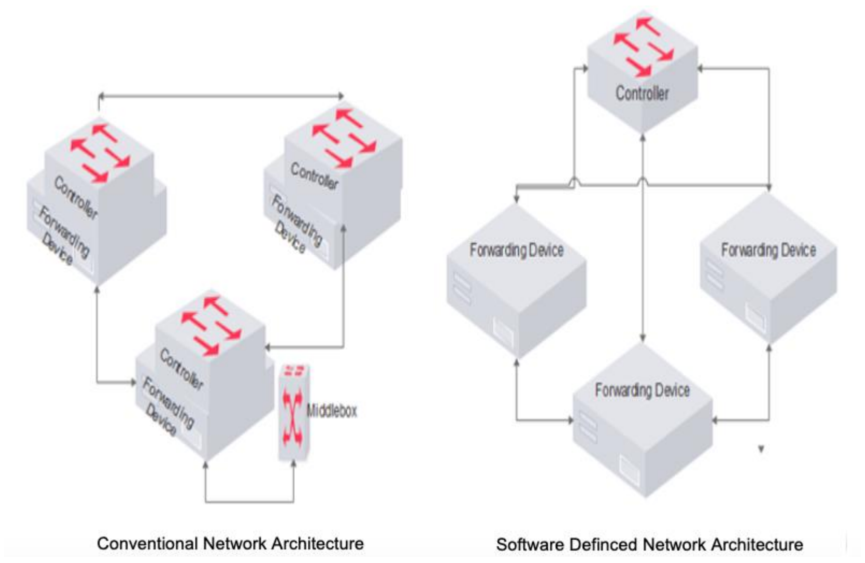


Figure 1. Conventional network vs software defined network

2. SOFTWARE DEFINED NETWORKING

SDN [10-13] has the ability to control, change, and manage network performance dynamically through software. This is achieved through open interfaces in contrast to the closed and proprietary defined boxes we have today. The SDN agenda allows for the centralization of network control. The centralized control embeds all the intelligence and sustains a network-wide view of the data path elements and nodes connecting them. This network-wide allows for easy modifications of network functions through the central. Figure 2 shows an SDN architecture.

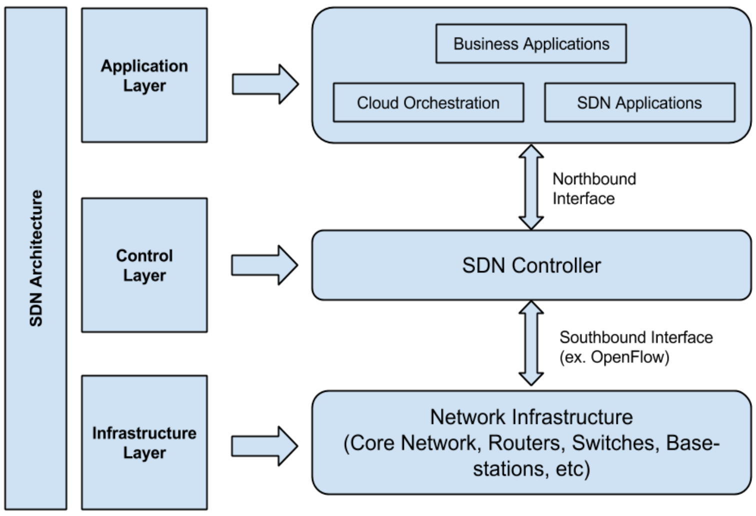


Figure 2. SDN architecture

2.1. SDN fundamentals

The SDN technology is built on four key tenets; First is on CP-DP separation, second on a logically centralized controller for the network-wide view, third for open interfaces between CP and DP devices, and lastly, the use of external application to support the programming feature of the network [13]. SDN consists of three layers and other APIs. The different layers have been discussed in [14] and APIs in [10, 15, 16].

a. Data Plane: The SDN DP are forwarding devices that are interconnected through wireless or wired communication channels. They are merely responsible for carrying user data across the network according to the path defined by the CP.
b. Control Plane: Forwarding devices are programmed by CP elements. The CP can therefore be seen as the network brain. All control logic rests in the applications and controllers, which form the CP.
c. Northbound Interface: This is an API that essentially takes the network requirements from the SDN applications and negotiates with the network controller that is responsible for providing applications with optimal network resources and paths.
d. Southbound Interface: this API defines the communication protocol between forwarding devices and the CP elements. This protocol formalizes the interaction between the CP and DP elements.

3. OPENFLOW NETWORK ARCHITECTURE

OF enabled network is different from conventional network infrastructure network by the decoupling of CP and DP layers. The Network Operating System (NOS) [13, 16, 17] is a layer in-between OF-enabled switch and a user-defined application that hosts the CP in an SDN. A communication between NOS and OF-enabled switch to notify the network events is done with the help of OpenFlow Protocols (OFP). A detailed architecture shown in Figure 3 and types of OFP is described in [6]. As illustrated in Figure 3, the key elements of the OF technology are the flow tables installed in OF switches, a controller installed in a remote host machine, and an OF secured protocol for the controller to communicate with switches. The OF method offers a flexible ability for real-time addition and update of numerous roles in the DP.

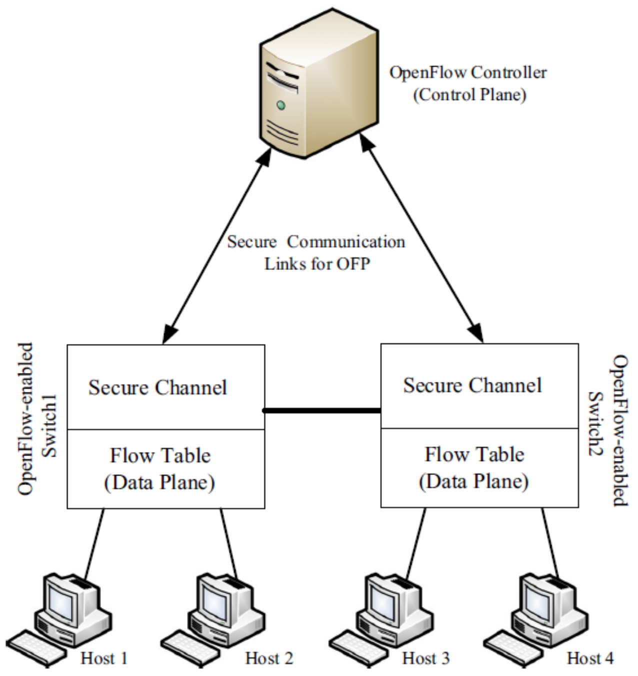


Figure 3. OpenFlow network architecture

In a conventional network architecture, when a packet arrives at the switches or routers, the packet will be processed using the info from header field and thereafter, route back such data to a dedicated link. The case is not the same for OF networks, as all the processing task is done by a centrally controlled unit and all switches only perform data forwarding by looking up their respective flow tables [17]. On the arrival of a packet, an OF enabled switch starts correlating the fields from the first flow table and moves on to the next using the flow table entry as described in [6]. If there is a correlation, then the specific actions are executed as defined by OF controller. One of the aims of centralized control of flows is to improve the speed of data communication. Traditionally, in any arrival of a data packet to a networking switch, the switch starts processing by executing initial networking protocols to resolve source and destination addresses. This task takes time for execution and resolution of a dedicated link. But in an OF, anytime data packet arrives at a switch, a controller resolves the address and simultaneously makes flow entries in a flow table of the switch [6]. Now, whenever another data packet arrives at the same switch, it will directly go through its flow table and forwards the data flow to a dedicated link, which not only saves time but also improves the data communication efficiency.

4. EXPERIMENTAL SET-UP

The experimental set up used for this work will be described in this section. This work makes a case for SDN as the future of networking by comparing the performance of OF-based network and conventional networks in terms of packet latency and jitter. The purpose of this experiment is to show that the SDN architecture design has an overall performance in terms of minimum jitter and average latency compared to the conventional network. A custom topology designed for both networks under consideration and a centralized controller placement approach was adapted to test the concept.

4.1. Custom network topology

There are two basic types of control model supported by SDN, each having its own different specialty based on the infrastructure and requirements. They are the centralized and the distributed controller architecture. There is also a hybrid control model which combines the advantages of the centralized and the distributed control models. In this work, the centralized control model has been considered as a proof of concept. The centralized control model entails using a single central controller to manage and supervise the entire network. In this model, the intelligence relies on a single decision unit. All the routing related information such as statistics, errors, and faults are collected and communicated to the controller with the help of OF.

An example of a custom centralized network control model used in this work is shown in Figure 4. From Figure 4, a host node sends a packet to the forwarding device and then the forwarding device sends it to the controller. This packet is processed and the flow table is updated. The controller only processes the initial packet of any flow from any forwarding device and updates the flow table accordingly. Depending on this flow table entry, all other packets of the same source get similar treatment to reach the desired destination. This flow rule is set by the controller and the forwarding device has no ability to update them without any instructions from the controller. One of the benefits of the SDN is that the forwarding device is free from any computation related tasks. However, this central control model has its own issues, which is not limited to a single point-of-failure, reliability, and scalability. An alternate approach is to make the control plane in a distributed fashion to execute applications of a centralized controller. In a network architecture with more than one CP, the distributed architecture could still have a central control unit through the logically centralized CP [13, 18-20].

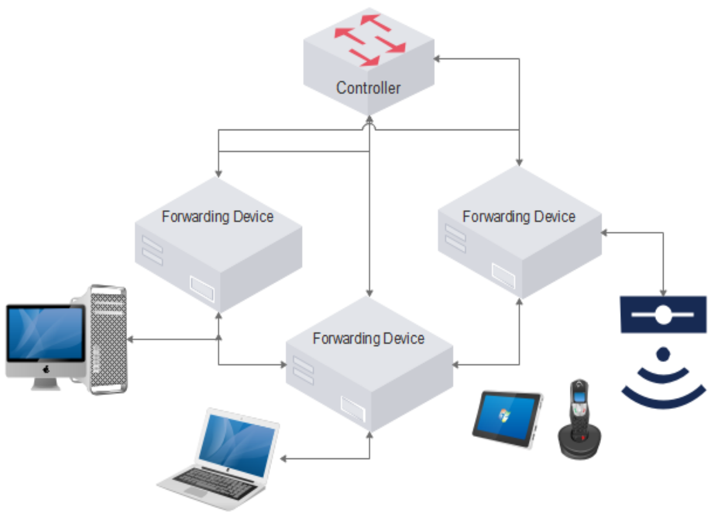


Figure 4. Custom network model of SDN

4.2. SDN architecture emulation tool

Mininet is a network emulation tool created by a group of researcher to be used as a tool for development, teaching and research in network technologies. It creates a realistic virtual network, running real kernel, switch and application code on a single machine. Mininet [2, 6, 15, 21-23] has gained wide popularity because first, only a few network emulation software exist for the purpose of implementing SDN standard. In addition, you can easily interact with your network using the Mininet Command Line Interface (CLI). Mininet works on various platforms like Windows, Linux, and MAC. For this work, all the experiments conducted was done using a computer with the following specifications as shown in Table 1.

Table 1. Computer system specification

Computer	Configuration
HP Pavilion Desktop	Linux operating system Ubuntu 16.04 64 bits 8 GB RAM 2GB VGA RAM 1000 GB of hard disk space Intel ® Core ™ i5-7400 CPU 2.4GHz processor Mininet and Mininet Controller (POX9, NOX10, Beacon12, FloodLight13, Maestro etc.) preinstalled Java/Python language support

Mininet has the capabilities that enable researchers or network programmers to create software-defined network prototype in a simple manner. The created prototype could be implemented into hardware for testing and validation. Some of the basic command syntax in Mininet and their functions as used in this work are explained in Table 2.

Table 2. Basic command syntax in Mininet and their functions as used in this work

Command Line	Description
	Creating a Network
sudo mn—topo single, 3—mac –switch ovsk –controller remote	A network can be created in MINET with a single command; this command creates a virtual network of three switches connected linearly and three virtual hosts, each host configured to the corresponding switch.
	Interacting with a Network
Mininet> h1 ping h2	In Minimet the entire virtual network can be controlled and manage from a single console, command is used to test connectivity between hosts. For example, test the connectivity between host h1 and h2. It will keep checking the connectivity between hosts until we stop the command
Mininet> h1 ifconfig -a	To show the host's h1-eth0 and loopback (Io) interface
Mininet> pingall	Alternatively, you can test the connectivity between all hosts by typing this command, it checks the connectivity/reachability among all hosts in the network.
mininet> xterm h1 h2	Another way to run interactive command and watch debug output is to run xtrem for one or more hosts. The command xterm h1 h2 open the exterm terminal for the host h1 and h2. Enter command in xterm of h1 i.e. ping "IP address assigned to h2". It will start checking the connectivity between host h1 and h2. Similarly, we can ping the host h1 from h2 using command in h2 xterm terminal ping "IP address assigned to h1."
Sudo mn -x	Alternatively, to start mininet xterm display option: To open xtrem display in Mininet, use command as follows; this is the basic xterm display command when we run this command on Mininet it startup xterm display host h1, h2, switch s1, and controller c0.
	In order to see the help menu
Sudo mn -h	To see a list of available commands
Mininet>help	Alternatively, this command shows various options which can write on Mininet CLI.
Mininet>nodes	Shows the nodes available in the current network of Mininet.
Mininet>dump	Shows the dump information about all nodes available in the current Mininet network.
Mininet>Dpctl	To control and edit flow tables.
Mininet>Iperf	To run TCP speed test.
sudo mn -c	To clear or clean up Mininet
Mininet>exit	To Exit Mininet

5. PERFORMANCE EVALUATION AND DISCUSSIONS

In order to evaluate the performance of the SDN architecture over the conventional networking architecture, the system has been emulated and the results are analyzed. The traditional way of measuring network performance is packet latency and jitter. Jitter is the amount of variation in latency/response time, in milliseconds. Jitter can be approximated as the difference between the maximum packet latency and minimum packet latency over a given period of time. Network latency is the term used to indicate any kind of delay that happens in data communication over a network. High latency creates bottlenecks in any network communication. It prevents the data from taking full advantage of the network pipe and effectively decreases the communication bandwidth. The impact of latency on network bandwidth can be temporary or persistent based on the source of the delays. The emulation for SDN is done using Mininet as previously described and GNS3 emulator was used for conventional networking. In this work, the time difference between consecutive sequence was also analyzed. The performance mainly depends on the hardware components like the number of processors, memory, hard disk and the network connection between the nodes.

To measure jitter, we take the difference between samples, then divide by the number of samples. If there is a change in network conditions (for instance, there is a sudden burst on the network and queues start to build on an interface) then this jitter will increase. The transit time between two consecutive packets will not be the same anymore: the second packet will have to go through a longer queue, spending more time, and generating positive jitter. Once this burst is over, the queue will progressively reduce, reversing the situation. Out of two consecutive packets, the second one will spend less time in the queues, and will therefore generate negative jitter. In a network, latency measures the time it takes for some data to get to its destination across the network. It is usually measured as a round trip delay. That is the time taken for information to get to its destination and back again. The round-trip delay is an important measure because a computer that uses a TCP/IP network sends a limited amount of data to its destination and then waits for an acknowledgment to come back before sending any more. Thus, the round-trip delay has a key impact on the performance of the network. Latency is usually measured in milliseconds (ms).

5.1. Conventional network architecture

A network set up shown in Figure 5 was designed in GNS3. After the network design, the latency of the packets can be evaluated by conducting a ping test between the hosts. Figure 6 shows the latency and jitter analysis of the conventional network model. From Figure 6(a) the results of the latency for packets sent from host H1 to host H2 is either same or more. Normally, at the point the first packet sent arrives at the switch, the switch checks its MAC table for the corresponding destination address in the MAC table. Since this is the first packet and there is no entry for the address, the switch forwards the packet to the router for the routing decisions. The router makes the routing decisions and sends the routing decision entry to switch. The switch forwards the packet accordingly and flushes the entry. This process is repeated for all packets that arrive at the switch. Therefore, this account for the high latency experienced by all the packets. From Figure 6(b), the time variation in latency/response time in milliseconds has been shown. This excessive jitter makes the service unusable by negatively impacting service quality. This is not acceptable and for a large-scale network where packets will travel through multi-hops.

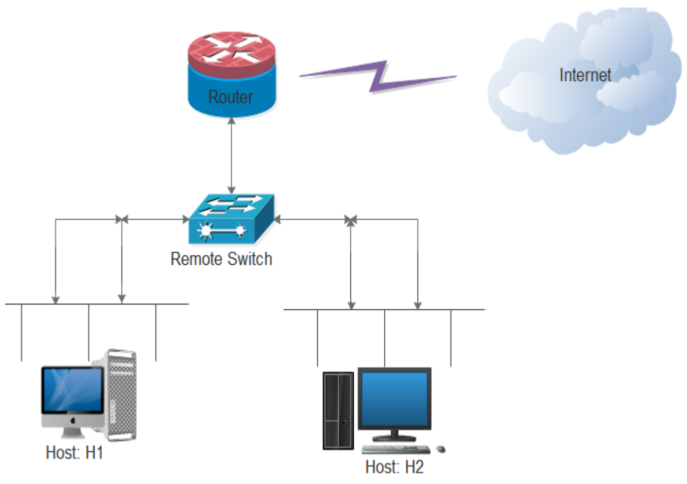
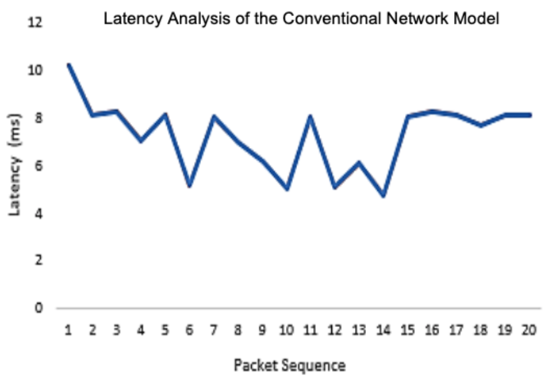
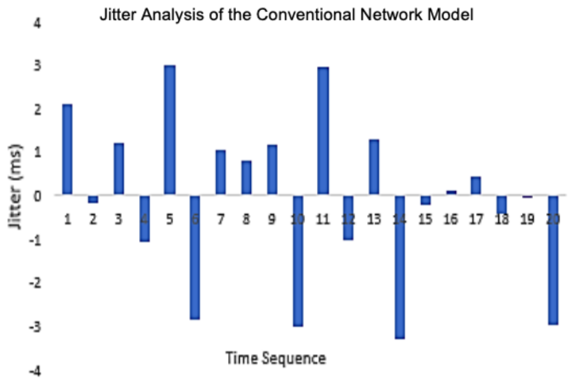


Figure 5. Set up of the conventional network architecture as emulated in GNS3



(a)



(b)

Figure 6. Performance of the conventional network model, (a) Latency, (b) Jitter

5.2. Software defined network architecture

Mininet, as explained in previous section, provides a comprehensive environment for prototyping SDNs. Mininet was used to create the set-up of the network as shown in Figure 7. The topology shown in Figure 7 is created using a single line command described in Table 2.

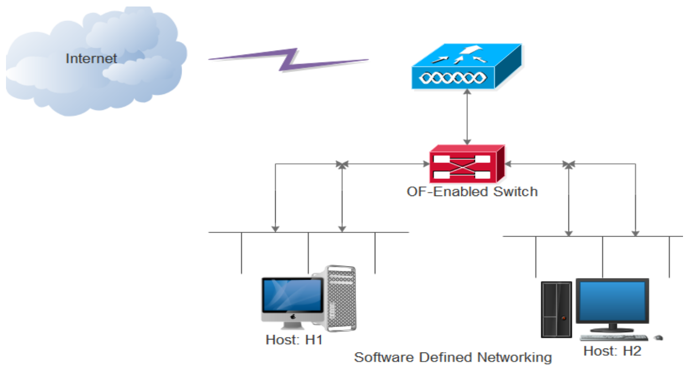
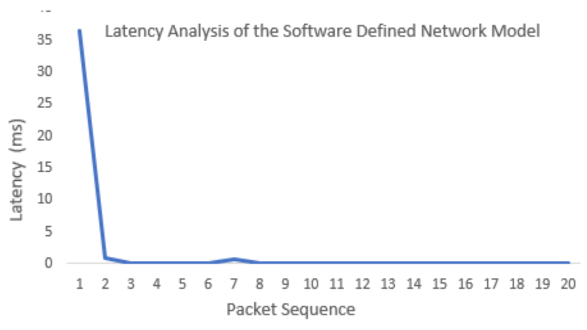
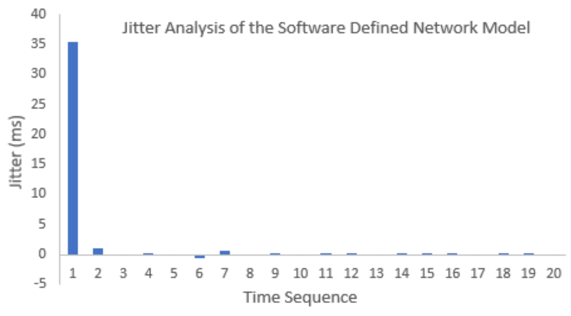


Figure 7. Set up of the conventional network architecture as emulated in mininet

As shown in Figure 8(a), the first packet takes 35.48ms. The time taken by the first packets in SDN is more compared to the consecutive packets, consecutive packets take much less time compared to the first packet in SDN. The reason for the time taken by the first packet in comparison to other packets is that the routing decision happens only for the first packet of a flow. Once the controller updates itself with the flow rule for the first packet, the switch buffers the flow rule in its flow table after thirty seconds [13, 24]. This accounts for why the time variation in latency/response time (jitter) were low or the same as shown in Figure 8(b). It eliminates the necessity of contacting the controller for routing decision for every packet, thus reducing the latency of the consecutive packets after the first packet. The consecutive packets are forwarded by the switch without contacting the controller for the routing decision. After thirty seconds, the buffer is timed out, the flow table is cleared the same procedure repeats.



(a)



(b)

Figure 8. Performance of the software network model, (a) Latency, (b) Jitter

5.3. Comparison of conventional and SDN network latency results

A comparison of the average latencies for the Tradition network and SDN is shown in Table 3. From Table 3, for the conventional network architecture, the minimum RRT for packets sent is 4.749ms with an average RRT of 7.256ms and a maximum RRT of 10.111ms. While for SDN architecture, the minimum RRT is 0.040ms with an average RRT of 1.969ms and a maximum RRT of 35.507ms. SDN has an overall performance in terms of minimum and average latency compared to the conventional network. With an average latency of over 3x lower than that of the conventional network, SDN can be seen as the future of networking.

Table 3. Comparison of the average latencies for the conventional network and SDN

Network Model	Minimum Latency	Maximum Latency	Average Latency
Conventional Network	4.749ms	10.111ms	7.25605ms
SDN	0.040ms	35.507ms	1.96940ms
Difference	4.709ms	25.396ms	5.28665ms

6. CONCLUSION

SDN is an interesting paradigm in networking that will change how network is built designed and operated. SDN promises networks that are open, proprietary independent, flexible and programmable. It will create new revenue opportunities for the network service providers through the creation of software-based applications. As far as the future of networking is concerned, the deployment of SDN can be designed efficiently to become faster, resilient, scalable, dynamic, reliable and most importantly secure. In this paper, the performance of SDN was analyzed by network connectivity test between hosts in an SDN architecture and a conventional network architecture. On the basis of results obtained, it is concluded that the performance of OF-enabled network with an average latency of over 3x lower is better than the conventional network. Ultimately, SDN improves network efficiency by reducing network overheads generated from frequent communication attempt between the CP and DP every time a packet is received. In SDN, the use of one CP architecture may not guarantee optimal results and also represent a single point of failure. Hence, the use of distributed CP architecture is recommended, although it also has its own peculiar challenges which includes how to place the controllers to guarantee optimal performance.

REFERENCES

[1] G. Wang, T. Ng, and A. Shaikh, "Programming your network at run-time for big data applications," in *Proceedings of the first workshop on Hot topics in software defined networks ACM*, 2012, pp. 103-108.

[2] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and Ligia Rodrigues Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, Bogota, 2014, pp. 1-6.

[3] A. L. Valdivieso Caraguay, L. I. Barona Lopez and L. J. Garcia Villalba, "Evolution and Challenges of Software Defined Networking," *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, 2013, pp. 1-7.

[4] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks: ACM*, 2010, p. 19.

[5] C.-C. Lo, H.-H. Chin, M.-F. Horng, Y.-H. Kuo, and J.-P. Hsu, "a flexible network management framework based on OpenFlow technology," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: Springer*, 2014, pp. 298-307.

[6] I. Z. Bholebawa, R. K. Jha, and U. D. Dalal, "Performance analysis of proposed OpenFlow-based network architecture using mininet," *Wireless Personal Communications*, vol. 86, no. 2, pp. 943-958, 2016.

[7] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74-98, 2014.

[8] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand and L. S. Leong, "Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) based Cloud infrastructure," *2014 2nd International Conference on Information and Communication Technology (ICoICT),* Bandung, 2014, pp. 505-510.

[9] N. McKeown et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.

[10] S. Hangloo and S. Kumar, "Software-Defined Networking: A Survey," *Computer Networks*. 2018.

[11] N. McKeown, "Software-defined networking,*" INFOCOM keynote talk*, vol. 17, no. 2, pp. 30-32, 2009.

[12] H. Kim and N. Feamster, "Improving network management with software defined networking," in *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114-119, February 2013.

[13] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[14] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453-471, 2014.

[15] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, Kuala Lumpur, 2015, pp. 205-210.

[16] F. Hu, Q. Hao and K. Bao, "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181-2206, Fourthquarter 2014.

[17] A. Lara, A. Kolasani and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493-512, First Quarter 2014.

[18] C.-S. Li and W. Liao, "Software defined networks," *IEEE Communications Magazine,* vol. 51, no. 2, pp. 113-113, 2013.

[19] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: a retrospective on evolving SDN," in *Proceedings of the first workshop on Hot topics in software defined networks: ACM*, 2012, pp. 85-90.

[20] Y. Kanaumi, S. Saito and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area OpenFlow-based network," *2010 International Conference on Network and Service Management*, Niagara Falls, ON, 2010, pp. 330-333.

[21] M. Sood, "SDN and Mininet: Some Basic Concepts," *International Journal of Advanced Networking and Applications*, vol. 7, no. 2, p. 2690, 2015.

[22] N. Pallavi, A. Anisha, and V. Leena, "Detection of Incongruent Firewall Rules and Flow Rules in SDN," in Artificial Intelligence and Evolutionary Computations in Engineering Systems: Springer, 2017, pp. 13-21.

[23] D. Kumar and M. Sood, "Software Defined Networking: A Concept and Related Issues," *International Journal of Advanced Networking and Applications*, vol. 6, no. 2, p. 2233, 2014.

[24] F. Souad and M. Moughit, "Evaluation of MTCP over POX Controller." *International Journal of Scientific & Engineering Research*. December 2016, vol 7 no. 12, 1079-1086.

BIOGRAPHIES OF AUTHORS

Paulson Eberechukwu Numan received his Bachelors in Engineering degree (Electrical and Electronics) in 2012 from the Federal University of Technology Akure, Ondo state, (Nigeria's topmost University of Technology). He obtained his Master of Engineering degree (Telecommunications and Electronics) in 2017 with a distinction and a recipient of the Best Post Graduate Student Award (BPGSA) from the prestigious Universiti Teknologi Malaysia (UTM), Johor, Malaysia. He is currently pursuing the Doctor of Philosophy Degree with the School of Electrical Engineering, Faculty of Engineering, UTM. His general research interest lies in the field of wireless communications, computer networking system design and optimization. In particular, he is interested in Software Defined Networks (SDN), Internet of Things (IoT), Cognitive Radio Networks (CRN), and Wireless Sensor Networks (WSN).

Kamaludin Mohamad Yusof received the B.Eng degree in Electrical-Electronics Engineering and M.Eng degree in Electrical Engineering from Universiti Teknologi Malaysia. He received Ph.D degree from University of Essex, U.K. He is currently a senior lecturer in the Division of Communication Engineering, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia and member of Advanced Telecommunication Technology. His current research interests include Internet-of-Things, Big Data and Software-Defined Network.

Muhammad Nadzir Marsono received the B.Eng. in Computer Engineering in 1999 and M.Eng. in Electrical Engineering in 2001, from Universiti Teknologi Malaysia. He received Ph.D. in Electrical and Computer Engineering from the Dept. of ECE, University of Victoria BC Canada in 2007. His current research interest include system level integration, multi-processor system-on-chip, network-on-chip, network algorithmics, and network processor architectures.

Mohd Husaini Mohd Fauzi received the B.Eng in Computer Engineering in 2011 and M.Eng. in Electrical Engineering in 2015, from Universiti Teknologi Malaysia. He is currently pursuing the Doctor of Philosophy Degree with the School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. His current research interest includes Wireless Sensor Network (WSN), Internet of Things (IoT), Fog Computing, Computer Networks and Systems, Localization and Tracking, Network Performance and Software define Network.

Salawu Nathaniel received the B.Eng. and M.Eng. degree from Federal University of Technology, Minna, Niger State, Nigeria in 2002 and 2010 respectively. He got his Ph.D in Electrical engineering from the Universiti Teknologi Malaysia. His current interests include wireless communication systems, cellular communication networks, handover issues in wireless communication networks.

Prof. Elizabeth N. Onwuka obtained a Bachelor of Engineering (B.Eng.) Degree from Electrical and Computer Engineering Department, Federal University of Technology (FUT) Minna, Niger State, Nigeria, in October 1992; a Master of Engineering (M.Eng.) Degree, in Telecommunications, from Electrical and Computer Engineering Department, FUT, Minna, Niger State, Nigeria, in March 1998; and Doctor of Philosophy (PhD) Degree, in Communications and Information Systems Engineering, from Tsinghua University, Beijing, People’s Republic of China, in June 2004. Her research interest includes Mobile communications network, Mobile IP networks, Handoff management, Paging, Network integration, Resource management in wireless networks, spectrum management, and Big Data Analytics.

Muhammad Ariff Baharudin received his B.Eng degree in Electrical-Mechatronics Engineering and M.Eng degree in Electrical-Electronics and Telecommunications Engineering from Universiti Teknologi Malaysia and also M.Eng degree in Communications Engineering from Shibaura Institute of Technology (SIT), Japan. He received his Ph.D degree from SIT, Japan. He is currently a senior lecturer in the Division of Communication Engineering, School of Electrical Engineering, Engineering Faculty and a member of the Advanced Telecommunication Technology (ATT) research group at Universiti Teknologi Malaysia. His current research interests include Internet-of-Things, Big Data, Connected Vehicles, Mobile Computing and Localization.