Developing an Asynchronous Technique to Evaluate the Performance of SDN HP Aruba switch and OVS

Ameer Mosa Al-Sadi, Member, IEEE
Department of Computing & Immersive Technologies
University of Northampton
Northampton, NN2 6JD. United Kingdom
Department of Computer Engineering
University of Technology, Baghdad, Iraq
Ameer.Al-sadi@northampton.ac.uk.

Ali Al-Sherbaz, Member, IEEE, James Xue, Member, IEEE, Scott Turner, Member, IEEE
Department of Computing & Immersive Technologies
University of Northampton
Northampton, NN2 6JD. United Kingdom
{ Ali.Al-Sherbaz, James.Xue, Scott.Turner }
@northampton.ac.uk.

Abstract— **Developers of Software Defined Network (SDN) faces a lack of or difficulty in getting a physical environment to test their inventions and developments. That drives them to use a virtual environment for their experiments. This work addresses the differences between the SDN virtual environment and physical SDN switches, which leads to equip a more realistic SDN virtual environment. Consequently, this paper presents a precise performance evaluation and comparison of off-the-shelf SDN devices, HP Aruba 3810M, with Open Virtual Switch (OVS) inside Mininet emulator. This work examines the variability of the path delay, throughput, packet losses and jitter of SDN in a different windows size of the packets and network background loads. Our conducted experiments consider a number of protocols such as ICMP, TCP and UDP. In order to evaluate the network latency accurately, a new asynchronous latency measurement technique is proposed. The developed technique shows more precise results in comparison to other techniques. Furthermore, the work focuses on extracting the flow-setup latency, caused by the external SDN controller when setting flow rules into the switch. The comparison of results shows a dissimilarity in the behaviour of SDN hardware and the Mininet emulator. The SDN hardware exposed higher latency and flow-setup time due to extra resources of delay, which the emulator does not possess.**

Keywords—SDN; OpenFlow API; Mininet; OVS; HP VAN controller; HP Aruba switch; Latency; Throughput; Jitter; Losses.

I. INTRODUCTION

Today's networks are experiencing a tremendous transformation from the architecture of traditional networks to SDN. SDN allows automating the network management by virtue of the centralised programmable control layer, which administrates the network and its traffic in consonance with the application requirements. Whereas, the switches in the data plane act as a "simple" forwarder of flows according to the rules received from the controller via OpenFlow Application programming interface (API) [1].

SDN researchers encounter difficulties in access, reserve or justify the real SDN testbeds to conduct experiments. Fortunately, promising alternative, explicitly Mininet, is being integrated into popular software packages of the network, such as OVS, to comprise a reliable SDN emulator [2]. Mininet is not viable to reproduce the exact behaviour of SDN vendors because it uses software switches such as OVS. Also, Mininet has not matured enough to mimic or follow some industry standards [3]. Mininet interfaces, switches and hosts could be configured profitably to any desired specifications, such as path delay and bandwidth, with the intention of fitting the behaviour of the admiring vendor. Regarding optimising the realistic results of Mininet, its specification needs calibrating to match one of the industrial devices. Therefore, this work aims to identify the difference between the behaviour of one off-the-shelf SDN devices, HP Aruba 3810M [4], and OVS inside Mininet emulator [2]. The finding of this study provides the research community with useful information in conducting more realistic emulation. The contributions of this paper as below:

- Introducing the data-plane measurement metric: this paper measures the data path latency, throughput, jitter and packet loss rate under a variable range of windows size of packets and background traffic.
- Extracting the control-plane metric "flow-setup latency": beside the data-plane metric, it was essential to bring up the latency variation of setting a new forwarding rule in switch flow-table after arriving un-listed flow, called flow-setup latency in this work. Flow-setup latency extracted from the same circumstances of data-plane metric, which is described above.
- Developing an accurate asynchronous latency measurement technique: of utmost importance was properly executing the latency measurement using accurate and intimately familiar tools to persuade any dissenters for the results' accuracy. That is why this work develops a simple asynchronous latency measurement technique which is based on reliable ping utility to measure the SDN latency.

The remainder of this paper is structured as follows: Section II presents the background and related work. Section III contains a brief description of the asynchronous latency measurement technique. The testbeds description and methods are exposed in Section IV. An analytical comparison of the obtained results is discussed in section V. Finally, in section VI we conclude the work.

II. BACKGROUND AND RELATED WORK

To optimise the realism of the emulator results, both the physical and emulated SDN test-beds need to be evaluated, which demands unified and idealistic measurement tools for both test-beds. Although, there has been a significant amount of research in the field of network measurement, it is imperative to: (1) identify and adjust a suitable measurement technique which serves the aim of this evaluation (section I); (2) investigate the findings of other research; (3) exploit their results as additional evidence that verifies the accuracy of the result this work. Therefore, a comprehensive literature review of network measurement was performed to avail the purposes above. The related work was classified into three parts: (A) traditional measurement techniques and matrices; (B) evaluation of virtual switches and emulator; (C) evaluation of SDN.

A. TRADITIONAL MEASUREMENT TECHNIQUES AND MATRICES

In traditional networks, two-way latency can be obtained using: (a) ping (ICMP) [5]; (b) (SYN-ACK/ACK) packets of TCP three handshake [6]; (c) TCP Timestamp option [7]; (d) and two-way UDP packets with embedded timestamp of connected hosts [5]. Ping is a powerful tool to extract the end-to-end Round-Trip-Time (RTT). It is a very effective tool to measure the latency variance in the different windows size of packets. However, it gives a rough estimation for one-way latency [5]. Ping tools predominate in latency measurement research as could be notified in [8][5]. The second technique to calculate the RTT is based on the difference in timestamp between the SYN-ACK packet and ACK packet (second and third packets) of the TCP connection establishment on the callee side [6]. (SYN-ACK/ACK) technique is unlike ping in the ability to change the window size of packets. It can find the RTT only for the small window size of packets (the packets of TCP three handshake). Both techniques suffer the additional latency which resulted from the processing time in the second end-host [8]. The third method to obtain the RTT is enabling the TCP timestamp option in TCP header [7]. It provides an acceptable precision of transmission latency, but it produces an extra payload added to the real load of original traffic. Sometimes, the packet with enabled TCP timestamp terminates by firewall devices because it reveals the timestamp of communicated nodes, which is clearly described in [8]. Finally, using the two-way UDP with an embedded timestamp of connected hosts to detect the Round-trip Delay Time (RTD) or One-way Delay (OWD). Like the TCP timestamp option, it supplies an accurate latency but requires a customised benchmark which can test only the latency of UDP packets [9].

B. EVALUATION OF VIRTUAL SWITCHES AND EMULATOR

Open vSwitch was used to emulate the SDN network in Mininet emulator because it can provide a complete functionality of the OpenFlow switches [2]. Several works evaluated Mininet for different purposes using the common measurement tools. The works in [10][11][12][13] measured the RTT of Mininet using ping utility. Others evaluated the RTT with ping and bandwidth of links using Iperf tools [14][15]. Finally, the ping ICMP packets were used to indicate controller flow setup time, while the TCP, UDP packets were emulated to estimate application latency in the Mininet [16]. All pre-mentioned works missed defining the bandwidth limit of links inside Mininet, which created an unrealistic measurement. However, their results were a good start point to realise the behaviour of SDN network and Mininet emulator.

C. EVALUATION OF SDN

The logically centralised control of SDN adding new measurement fields such as the latency of the control layer. In SDN, the methods of determining the network latency would be handled by the data plane or by the control plane.

On the one hand, most works used the controller to compute the path latency in different ways. The authors in [17][18][19] sent probe packets from the controller which passed through the path back to the controller. The path latency extracted by subtracting the control path latency from the difference of sending and receiving timestamps. The work in [20][21][22] uses the same method except for the way of determining the path. The path is passively determined after collecting all the entries of the flow-table from the OpenFlow switches, then sending a probe packet from the controller to compute the paths latency. Another piece of research uses the looping technique by applying a loop of special service packets through the path with specified Time-to-live (TTL). OpenFlow switches on that path: (a) decrement TTL; (b) register number of iteration; (c) and forward the packet in the loop while the TTL is not zero, otherwise, forward it to the controller. The controller then calculates the latency using TTL and iteration number [23][24]. Another piece of research proposed the Queue Length Method [23]. It considers the processing, propagation and transmission delay as constant values and uses the detected queuing delay to estimate the path delay.

On the other hand, some researchers deny over-heading the controller with additional computation. Also, they noticed that the control layer latency varies more than the switch forwarding latency, which heavily affects the accuracy of latency measurement of the data path [20][25]. For example, [25] employs a monitoring host in data-plane to measure the path latency using an active probe packet. Also, [14] uses the traditional ping and Iperf tool to identify the latency and bandwidth of network in his study.

Finally, few studies focus on resolving the control layer latencies. [26] creates a python script to probe packets between OpenFlow switches and controller to dissect fractions of the control layer latency. [27] studies the effect of varying the load of the control plane on it is latency using background control probe packets and ping utility on the Pica8 physical switch. At last, High-Fidelity Switch Models for SDN Emulation [3] is the most related work to this paper. It tests and compares flow-setup latency of the HP ProCurve, Quanta, and Monaco physical switches with OVS. The OVS was emulated on Linux based virtual machine and not in Mininet. The test in [3] is limited to measure the control layer latency under different conditions while missing the measurement and comparison of data plane latency, While this paper measures the data plane latency on the physical and Mininet testbeds.

III. A BRIEF DESCRIPTION OF THE PROPOSED ASYNCHRONOUS LATENCY MEASUREMENT TECHNIQUE

Since the intent here is to extract actual path latency and not its approximation, ping utility might not fulfil the purpose by itself. For the reason that, the RTT which is computed by ping tool comprising an extra processing time which belongs to the callee side. Therefore, simple steps were added to optimise the latency which obtained from RTT of the ping tool, as follows:

1. Ping from sender to receiver as shown in figure1.
2. Capturing the send request at both sides to record the timestamp of:
   a) Send-Request-Time (SReqT) at the sender.
   b) Received-Request-Time (RReqT) at receiver.
3. Capturing the returned reply at both sides to record the timestamp of:
   a) Send-Reply-Time (SRepT) at receiver.
   b) Received- Reply -Time (RRepT) at the sender.
4. Compute processing time (**Δ2**) in the receiver side "callee side" by subtracting (SRepT) from (RReqT):

$$\Delta 2 = (SRepT) - (RReqT) \quad (1)$$

5. Compute the actual Latency from ping RTT in two steps:
   a) Calculate the ping RTT (**Δ1**):

$$RTT = \Delta 1 = (RRepT) - (SReqT) \quad (2)$$

   b) Calculate two-way-latency (**Δ3**):

$$\text{Two-way-Path-latency} = \Delta 3 = \Delta 1 - \Delta 2 \quad (3)$$

6. Compute the one-way-latency :

$$\text{One-way-latency} = \Delta 3/2 \quad (4)$$

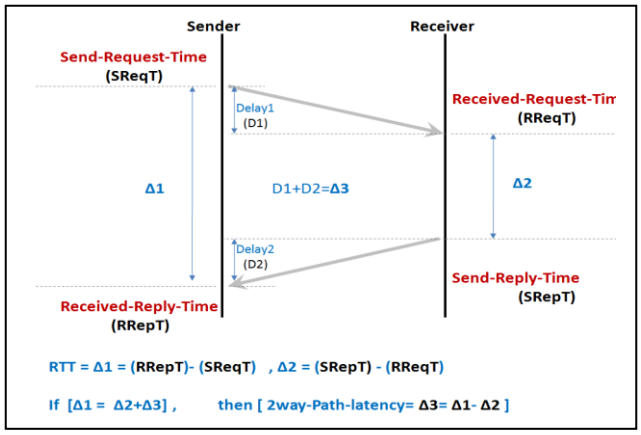

Fig.1. The proposed asynchronous latency measurement technique.

The (Δ3) represents two-way-latency through the switch with considering that the link delay is negligible. For the reason that the maximum propagation delay for 1 meter approximate to 5.5 ns [28].

One-way latency is half of two-way latency, while SDN provides the same path for the request and the reply packets. In that technique, it is possible to compute the two-way or one-way path latency accurately. Also, it demonstrates parity with one-way synchronised techniques without the severing of synchronising the end host or customising a software benchmark. Furthermore, it can be used to measure the accurate latency for any traffic such TCP, UDP, ICMP, etc.

IV. THE TESTBEDS DESCRIPTION AND METHODS

This section briefly describes the testbeds components and topology followed by the measurement methods.

A. Testbeds description:

The tests are performed in the SDN laboratory of the University of Northampton. Two testbeds were used to examine the physical and emulated SDN. The physical testbed comprises of: HP VAN SDN Controller 2.7.18 [29] which runs over a VirtualBox on a dedicated machine; physical HP Aruba 3810M SDN switch [4], which construct a single topology as shown in figure 2; Four computers represent the communicated hosts; and, one-gigabit connections. Meanwhile, the emulated testbed consists of the same controller mentioned above and Mininet emulator, which hosted on the single server. Mininet emulated single topology as well with FastEthernet speed connection 1000Mbit/s, figure 2.

All machines are running Windows 8.1, 64-bit with an Intel Core i7 CPU and 16 GB of RAM. Additionally, the server which hosted the controller and Mininet has an SSD hard drive.

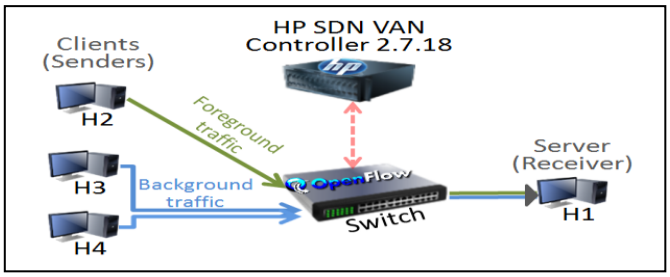

Fig.2. The testbed Structure.

B. Measurement methods:

The Measurement method is divided into three types of measurement; they are "Latency", "Throughput" and "Jitter and packet losses" measurement. Every measurement type has several tests which will be described below. For studying the impact of background traffic on foreground traffic, every test performed three times. The first time the test measured the performance of SDN only with foreground traffic between H2 and H1, see figure 2. In the second and third times, the foreground traffic was tested after saturating the network with background traffic, where both H3 and H4 sent background traffic to H1 with 250 and 375 Mbit/s. That saturated the network with total background traffic of 500 and 750 Mbit/s for the second and third times respectively. Next will be a brief demonstration of three measurement types and their tests:

1) Latency measurement. Firstly, path latency was measured using: (a) the ping utility; (b) the proposed

asynchronous latency measurement technique; (c) and (SYN-ACK/ACK) measurement technique, which showed in section II.A. The results of these three techniques were compared to show the accuracy of the proposed measurement technique. In ping and asynchronous latency measurement technique, the test performed for three windows size of packets 1.5, 10 and 65 KBytes. The three windows size help to discriminate SDN behaviour for small, medium and large application packets. Whereas, for (SYN-ACK/ACK) measurement technique, the test performed only for a small window size of packets because the window size always starts small through the three handshake process of TCP connection. The results represent the average value of RTT for twenty pings or TCP probe packets.

Secondly, the flow-setup latency found from the RTT of the first packet of each flow. It was extracted from ping and the proposed asynchronous technique as same as the procedure mentioned above.

2) Throughput measurement. The throughput was tested for TCP and UDP traffic with four different windows size of the packets. This time, the test was done with 1.5, 10, 65 and 150 KByets as a window size for foreground traffic. The reason for adding a window size of 150 KByets was that the UDP traffic was not able to achieve the maximum throughput with a window size of 65Kbytes on HP Aruba switch, see figure 5-b. The results averaged the throughput of a twenty TCP/UDP connection transmitted for 120 seconds using Iperf tool.

3) Jitter and packet losses measurement. Both of them were obtained for UDP traffic with four different windows size of the packets, similar to the throughput measurement. The results calculated the average of twenty UDP connection lasted for 120 seconds using Iperf as well.

V. RESULTS EVALUATION

This section will discuss the results which were obtained from the practical implementation of experimental traffic on real components.

A) Path latency. The proposed asynchronous technique (figure 3-b) showed more accurate results than ping measurement (figure 3-a). The proposed technique shows lower latencies values which increased linearly with the increment of network background load, while ping shows higher and unstable latencies. The SYN-ACK/ACK technique (figure 3-c) showed near results to the results of the proposed technique but only for a small window size of packets. That verified the accuracy of the results of this work.

In the results of the proposed technique, HP Aruba switch occupied latency (red-line) ten times larger than OVS latency (green-line) for the small window size of packets (1.5 KByets). Meanwhile, this difference reduced to four-times of latency in a window size of 10 KByets and became only two times the latency for a window size of 65 KByets.

B) Flow-setup latency. The flow-setup latency of the small window size of packets and zero background load was around 280ms for HP Aruba switch (red-line) and 3ms for OVS (green-line), when computed by the proposed asynchronous technique, see figure 4-b. Diversely, flow-setup latency got around 400ms for HP Aruba switch (red-line) and 3.5ms for OVS (green-line), if extracted from ping RTT, see figure 4-a. The difference in latency between the two techniques resulted from removing the processing latency of the second end from the flow-setup latency. That is proving the results validity of the asynchronous latency measurement technique specifically for the physical testbed.

Additionally, it is feasible from the figure 4 that, the difference in flow-setup latency between the real network and Mininet emulator is very big. The latency variation ranges from 100 to 40 times of flow-setup latency for small to the large window size of packets. Finally, all latencies increased with the rising of the background traffic of the network.

C) Throughput. As displayed in figure 5, physical and emulated SDN provide different throughput. However, the links of both testbeds were configured with the same bandwidth. On the one hand, HP Aruba generates very poor TCP throughput with a small window size of packets while OVS presents higher TCP throughput. On the other hand, HP Aruba provides best TCP throughput with a large window size of packets.

The throughput of UDP traffic changed a lot in HP Aruba switch according to the window size of packets while a small variation of UDP throughput occurs in OVS. TCP/UDP throughput degraded when the background traffic of network is boosted.

D) Jitter and Packet Losses rate. The tests showed that OVS possess a lower jitter and packet losses rate than HP Aruba switch. However, the jitter and packet losses rate of HP Aruba are more responsive to the change of window size of packets and network background traffic, see figure 6.

VI. CONCLUSION

This paper reveals that there is a gap in the performance of the physical and emulated SDN testbeds. This difference in their behaviour needs to be identified precisely in terms of optimising the results realism of the emulator. Therefore, this work performed a comprehensive literature review about network measurement techniques to select the optimal techniques for accurate measurements. Also, the literature review discloses that the measurement techniques which employed the controller for monitoring demonstrate more efficiency for real-time and estimated latency measurement. Whereas, the measurement techniques which exploited the data plane provide more accurate measurement (see section II.C). After that, this work develops an accurate measurement technique in data plane namely, "Asynchronous Latency measurement technique", to study and compare the performance of the two testbeds. The findings of this study are as follows: the path latency of physical SDN is noticeably larger than the Mininet; the flow-setup latency is massively greater in physical devices; Mininet possesses less reactivity for the changes of the window size of packets and background load than the physical testbed regarding throughput, jitter and losses packet rate.

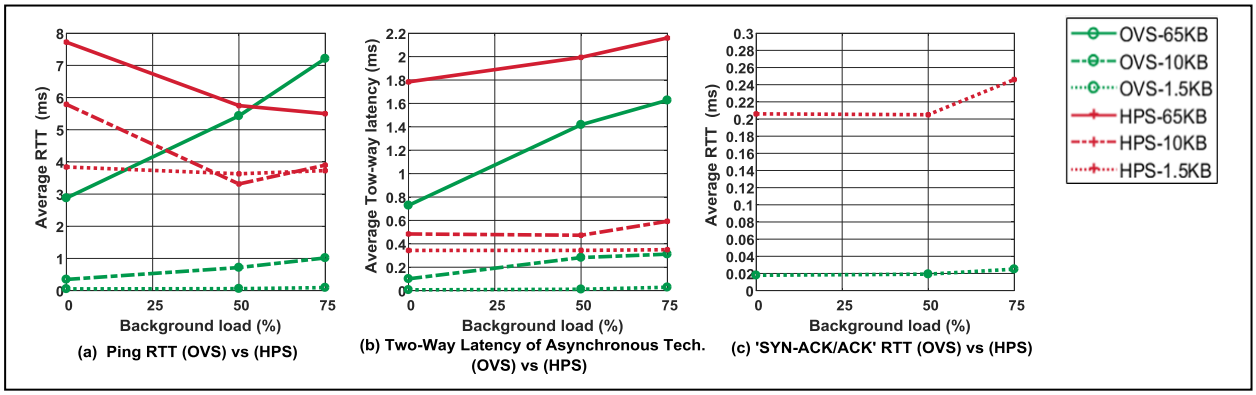

Fig.3.Two-Way Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in three different techniques.

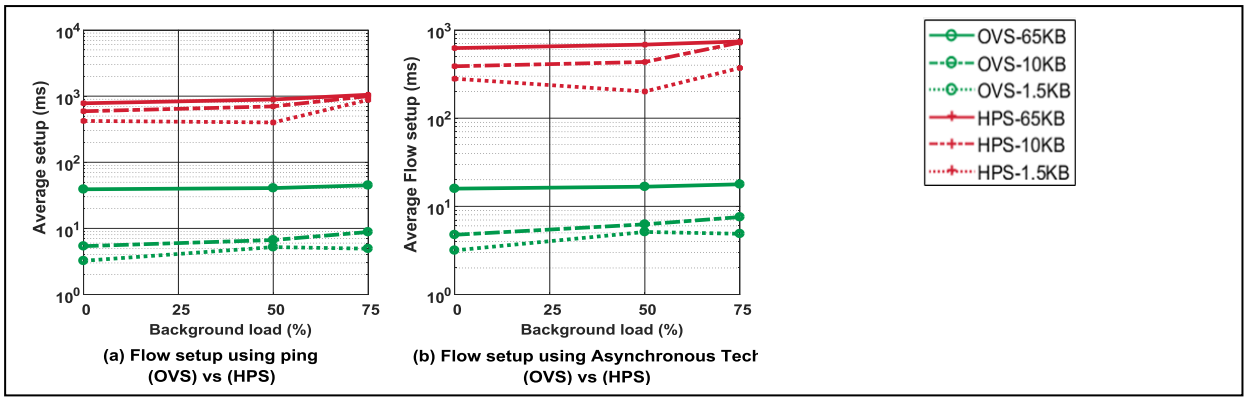

Fig.4. Flow setup Latency measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) in two different techniques.

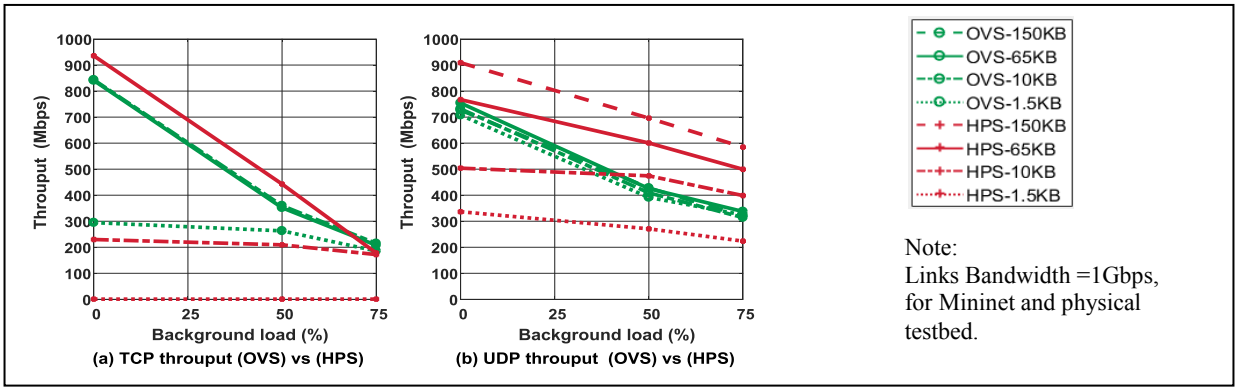

Fig.5. Throughput measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) for TCP and UDP traffic using Iperf.

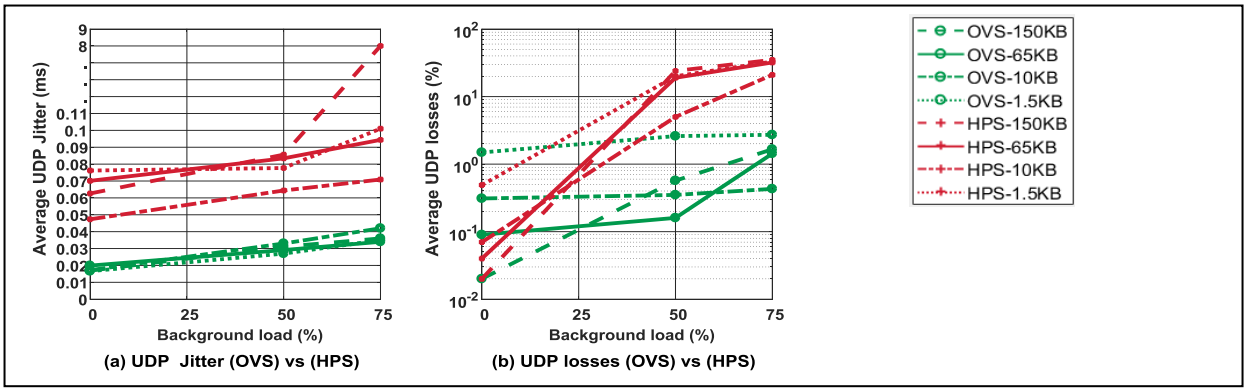

Fig.6. Jitter and losses packet rate measurement comparison between Mininet (OVS) and Physical testbed (HP Aruba) for UDP traffic using Iperf.

Fortunately, Mininet composes of configurable components. Therefore, this shortage in its performance could be overcome by calibrating this emulator with parameters which degrade the result differences from the physical SDN and optimise its results realistically.

The probable future work, this study could be extended to:

- Evaluate other SDN switches.
- Evaluate SDN switches from a different aspect such as the effect of application layer on switch performance.
- Optimising Mininet performance to match the behaviour of a specific vendor of physical SDN switches.

ACKNOWLEDGMENT

This work was supported in part by Iraqi Ministry of Higher Education and Scientific Research - scholarship no.21573 for the first author.

REFERENCES

[1] O. N. Foundation, "Software-Defined Networking (SDN) Definition - Open Networking Foundation," 2015. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition. [Accessed: 02-Dec-2014].

[2] Mininet Team, "Mininet Overview," 2016. [Online]. Available: http://mininet.org/overview/. [Accessed: 04-Jun-2016].

[3] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-Fidelity Switch Models for Software-Defined Network Emulation," *Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. networking. ACM,* 2013.

[4] H. Packard Enterprise, "Aruba 3810M Switches Installation and Getting Started Guide," 2016.

[5] J. Wang, M. Zhou, and Y. Li, "Survey on the End-to-End Internet Delay Measurements," Springer, Berlin, Heidelberg, 2004, pp. 155–166.

[6] Dhaval N. Shah, Virupaksh Honnur, Dalen, and D. Bosteder, "System and method for measuring round trip times in a network using a TCP packet," U.S. Patent No. 6,446,121., 2002.

[7] V. Jacobson, R. Braden, and D. Borman, "RFC 1323,TCP extensions for high performance," pp. 1–38, 1992.

[8] I. Prieto, M. Izal, E. Magaña, and D. Morato, "A Simple Passive Method to Estimate RTT in High Bandwidth-Delay Networks," in *The Seventh International Conference on Evolving Internet*, 2015.

[9] F. Sans and E. Gamess, "Analytical Performance Evaluation of Different Switch Solutions," *J. Comput. Networks Commun.*, vol. 2013, pp. 1–11, 2013.

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

[11] P. Danielis, V. Altmann, J. Skodzik, E. B. Schweissguth, F. Golatowski, and D. Timmermann, "Emulation of SDN-supported automation networks," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015, pp. 1–8.

[12] Gustaf Jan Gunnar Nilstadius, "Software defined networks with high availability," Czech technical university, Department of Telecommunication Engineering, 2016.

[13] S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, 2014, pp. 1–6.

[14] M. Koerner and O. Kao, "Evaluating SDN based Rack-to-Rack Multi-path Switching for Data-center Networks," *Procedia Comput. Sci.*, vol. 34, pp. 118–125, 2014.

[15] I. Zoher Bholebawa and U. D. Dalal, "Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet," *Int. J. Comput. Commun. Eng. 5.6*, p. 419, 2016.

[16] D. Turull, M. Hidell, and P. Sjodin, "Performance evaluation of openflow controllers for network virtualization," in *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*, 2014, pp. 50–56.

[17] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow," *Netw. Serv. Manag. (CNSM), 2013 9th Int. Conf. on. IEEE*, 2013.

[18] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–8.

[19] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang, and H. V Madhyastha, "Software-defined Latency Monitoring in Data Center Networks," *Int. Conf. Passiv. Act. Netw. Meas. Springer Int. Publ.*, pp. 360–372, 2015.

[20] S. Wang, J. Zhang, T. Huang, J. Liu, Y. Liu, and F. R. Yu, "FlowTrace: measuring round-trip time and tracing path in software-defined networking with low communication overhead," *Front. Inf. Technol. Electron. Eng.*, vol. 18, no. 2, pp. 206–219, Feb. 2017.

[21] K. Agarwal, E. Rozner, C. Dixon, and J. Carter, "SDN traceroute: Tracing SDN Forwarding without Changing Network Behavior," *Proc. third Work. Hot Top. Softw. Defin. networking. ACM*, pp. 145–150, 2014.

[22] sFlow.org, "sFlow: Software defined networking," 2012. [Online]. Available: http://blog.sflow.com/2012/05/software-defined-networking.html. [Accessed: 07-May-2017].

[23] D. Sinha, K. Haribabu, and S. Balasubramaniam, "Real-time monitoring of network latency in Software Defined Networks," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 2015, pp. 1–3.

[24] V. Altukhov and E. Chemeritskiy, "On real-time delay monitoring in software-defined networks," in *2014 First International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*, 2014, pp. 1–6.

[25] A. Atary and A. Bremler-Barr, "Efficient Round-Trip Time monitoring in OpenFlow networks," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[26] M. Kuniar, P. Perešíni, and D. Kosti, "What you need to know about SDN control and data planes," *No. EPFL-REPORT-199497*, 2014.

[27] J. Sonchack, A. J. Aviv, and E. Keller, "Timing SDN Control Planes to Infer Network Configurations," *Proc. 2016 ACM Int. Work. Secur. Softw. Defin. Networks Netw. Funct. Virtualization. ACM.*, 2016.

[28] G. Beckhoff Automation GmbH & Co. KG, "Infrastructure for EtherCAT/Ethernet Technical recommendations and notes for design, implementation and testing Table of contents," 2017.

[29] H. Packard Enterprise, "HPE VAN SDN Controller OVA: Free Trial | SDN App Store," 2017. [Online]. Available: https://marketplace.saas.hpe.com/sdn/content/hpe-van-sdn-controller-ova-free-trial. [Accessed: 07-May-2017].