OFBench: Performance Test Suite on OpenFlow Switches

Ying-Dar Lin, *Fellow, IEEE*, Yu-Kuen Lai, *Senior Member, IEEE*, Chen-You Wang, and Yuan-Cheng Lai, *Member, IEEE*

***Abstract*—Performance issues of OpenFlow switches are attracting a lot of attention owing to the potential large-scale deployment of software-defined devices. This paper presents the OFBench which is an automatic test suite for evaluating the performance of OpenFlow switches. The design, as part of the Automation Control Test System (ACTS) development, is based on a controller-agent architecture which allows the development of test cases that are written in a high-level script language. In addition to the end-to-end measurement methodology, novel methods are proposed to further profile the internal performance metrics, which are difficult to get due to the black-box nature of the device under test. The prototype of this suite currently comprises five test cases to evaluate five performance metrics, which are action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy. OpenFlow switches are evaluated and three issues are observed associated with switches during the testing. First, some switches may not be well implemented in the design of apply-action instructions. Second, some switches suffer from random crashes with a high volume of bursty packet-in traffic. Finally, the timer of idle-timeout is often not reset properly with matching flow entry.**

***Index Terms*—Computer networks, openflow protocol, performance evaluation, software-defined networks, system performance, testing.**

I. Introduction

SOFTWARE-DEFINED networking (SDN) is an emerging network architecture. The major difference between SDN and the traditional network is the decoupling of the control and data plane operation. Therefore, in SDN, the data plane needs to communicate with the control plane through a specific communication protocol. OpenFlow is a communication protocol standard [1] that is used by the controller to manipulate data plane operations. As a packet arrives at the OpenFlow switch, selected tuples in the packet header fields are examined according to the rules in the flow table. In the absence of matching, the packet is treated as a new flow and the table-miss operation is triggered. Depending on the configuration, either the packet is simply dropped or an OpenFlow message is sent to the controller that is waiting for further action. The controller may insert a new flow entry in the flow table with the matched packet. The flow entry is a set of data structure that is composed of such information as packet headers, counters, and instructions [2]. Therefore, these interactions with the controller make the operation of the OpenFlow data plane (OFD) very different from the traditional data plane (TD) [3].

However, to the best of the authors' knowledge, no performance test standard exists for the OpenFlow switch. Available performance test suites concentrate on the part of the TD [4]. Some open-source tools such as OFTest [5] and Ryu certification [6] are available for performing the OpenFlow switch conformance test. These tools only determine the interoperability [7] for OpenFlow switches under test. Spirent proposed a white paper on OpenFlow performance testing [8]. However, the white paper only addresses selected test cases and offers some general guidelines for the performance evaluation of OpenFlow switches.

Manuscript received November 4, 2016; revised March 23, 2017 and May 17, 2017; accepted June 10, 2017. This work was supported in part by the Ministry of Science and Technology, Taiwan, and in part by Estinet Technologies, Inc., and Chunghua Telecom. *(Corresponding author: Yu-Kuen Lai.)*

Y.-D. Lin is with the Department of Computer Science, National Chiao Tung University, Hsin-Chu 300, Taiwan (e-mail: ydlin@cs.nctu.edu.tw).

Y.-K. Lai is with the Department of Electrical Engineering, Chung-Yuan Christian University, Taoyuan 32023, Taiwan (e-mail: ylai@cnsrl.cycu.edu.tw).

C.-Y. Wang was with the Department of Computer Science, National Chiao Tung University, Hsin-Chu 300, Taiwan He is now with Proscend Communications, Inc., Hsin-Chu 300, Taiwan (e-mail: arthur81542@gmail.com).

Y.-C. Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Digital Object Identifier 10.1109/JSYST.2017.2720758

A. Motivations

Generally, an overall process of the OpenFlow transaction, abbreviated as D2C2D, can be further categorized into two parts: the *data-plane-trigger-control-plane* (D2C) and the *control-plane-trigger-data-plane* (C2D). Many studies [4], [9]–[12] have addressed these topics. However, most of them focus on TD testing without addressing all of the test metrics of OFD, D2C, and C2D. Since the processes of D2C and C2D are the core of the OpenFlow protocol, the suite for testing OpenFlow switch must address items, as discussed in Section III, that are related to the evaluation of the performance of the D2C2D process along with the OFD.

In addition to the switch performance evaluation, an accurate switch model [13] can be better augmented with the outcomes for simulation in order to understand the performance and scalability of a large software defined networks. Huang *et al.* [14] proposed high-fidelity OpenFlow switch models for accurate SDN emulation based on their latency measurement methodologies. Shang and Wolter [15] proposed an queuing model for the packet processing time of OpenFlow switch and controller. Sato *et al.* [16] proposed an abstract model of SDN architectures such that a globally optimized performance can be achieved.

For all of the proposed switch models, they all depend on the accurate modeling of performance parameters such as control latency, forwarding speed, and blocking probability [17] which are all associated with the interactions of control plane and data plane processing. However, measuring the internal performance parameters in the test cases is challenging, because the device-under-test (DUT) is effectively a black box. For example, the exact duration of each action time is difficult to measure because the processes comprise many steps. The accuracy of idle timeout for each flow entry is also difficult to measure because the timer is reset as the packets arrive and are matched.

Based on the white paper [8], this paper further extends test cases that address the most critical aspects of OFD, D2C, and C2D testing. As the ongoing efforts of the Automation Control Test System (ACTS) [18] development, this paper presents the selected test cases involving items, illustrated in Fig. 2, that are related to flow action, pipeline, packet-in, packet-out, and flow-mod operations.

The main contributions of this work are summarized below.

1) An automatic test suite is developed and implemented based on the controller-agent architecture. The suite can control remote agents to generate and analyze networking traffic. The agents yield results that are provided to the controller based on the proposed test cases.
2) The test suite can run on the commodity PCs without specialized hardware support.
3) This work proposes five methods in three categories to address issues of performance measurement including action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy.
4) The first category, called mirror-in-processing, is based on the apply-action instruction in OpenFlow to mirror packets in the process. The system can measure the processing time of each stage in the DUT based on the mirrored packets.
5) The second one, named calculated traffic, is based on the burst-until-loss and back-to-back traffic to estimate the buffer size and pipeline efficiency in the OpenFlow switch.
6) The third one, called masked entry, is based on the priority, match fields, and actions of flow entry to control the timer where two flow entries can be synchronized for proper measurement.

The rest of this paper is organized as follows. Section II presents the background and related work of software defined networks and OpenFlow protocol with emphasis on performance related issues and testing. Section III provides a statement of the problem of interest. Section IV introduces the proposed methods for mirror-in-processing, calculated traffic, and masked entry. Section V discusses the implementation in detail and presents relevant experimental results. Section VI draws conclusions and makes suggestions regarding future work.

II. Background and Related Work

OpenFlow, a communications protocol for SDN architecture, consists of communications messages and mechanisms that enable packet processing for OpenFlow switches and controllers. The packet processing mechanism is based on the flow policy, which is a combination of match fields and instruction sets. The flow policy is implemented as a flow entry and stored in the flow table.

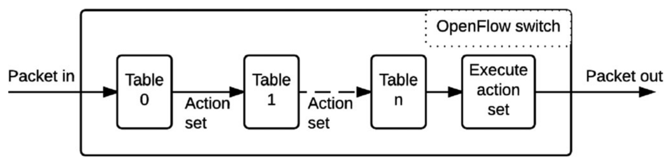

Fig. 1. Typical packet processing pipeline over multiple tables in an OpenFlow switch [2].

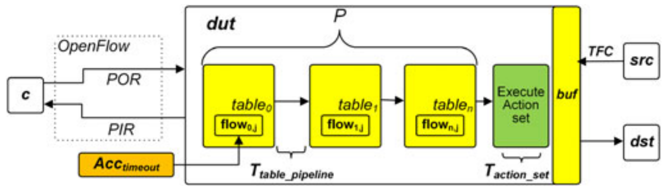

Fig. 2. OpenFlow performance testing for a given device under test. This work proposes methods to address the issues of performance measurement including action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy.

TABLE I
OpenFlow Instruction With Executing Priority

Instruction	Description	Executing Priority
Meter	Apply the specified metering to packet.	6
Apply-Actions	Apply actions immediately to packets.	5
Clear-Actions	Clear action set.	4
Write-Actions	Write actions to action set.	3
Write-Metadata	Write the masked metadata value.	2
Goto-Table	Execute the table pipeline.	1

The whole process for packet processing is illustrated in Fig. 1. As a packet arrives, the switch tries to match the flow entries in the table of multiple stages. Once the packet has been matched with the entry in the table, the action set is updated and possibly directed to other tables in a pipelined fashion. Finally, at the end of the pipeline process, the action set is executed. The flow entry, which is the most important part of the process, comprises priority, counters, match fields, timeouts, cookies, and instructions. The priority influences the order of executing of flow entries. The entries of timeouts contain timers of hard-timeout and idle-timeout values for removing a flow entry when it has expired. The match fields are a set of records consisting of packet headers from layer one to layer four. The instruction set contains operations to be executed on the incoming packet, which is matched with the match fields. Table I presents the available instructions and order of executions.

A. *Control and Data Plane Interactions*

Typically, the items in a performance test are chiefly associated with the part of TD that involves latency, loss, buffer size, throughput, and jitter. However, the most important factor in testing the performance of an OpenFlow switch is the interactions between the control plane and the data plane. The parameters associated with the testing of OpenFlow performance can be summarized in categories of C2D, D2C, and OFD. In C2D, many messages are sent from the controller to the switches. Therefore, the performance parameters that are related to this category are associated with the flow-mod, configuration, and state query of the switch. The switches send messages concerning some events back to the controller in D2C. These messages include packet-in, flow-removed, status report, and errors. Therefore, some informational contents in these messages should be treated as important parameters in OpenFlow performance measurement.

Additionally, in most of cases, the D2C2D processes are commonly executed upon the arrival of new flows. The D2C2D processes utilize the messages of packet-in, packet-out, and flow-mod. The processing rate of packet-in messages is an important indicator showing the effectiveness of OpenFlow switch handling the new type of flows. If the rate is low, then the OpenFlow switch is not able to process a large number of new flows in a short period. The packet-out message that is generated by the controller, is used to execute actions for the packet-in message in the D2C2D process for the first packet of a new flow. The processing rate of packet-out message sent by the controller reflects the ability of an OpenFlow switch.

Finally, the period for which the flow entry remains active is set by the flow-mod message that is sent to the switch. Therefore, processing rates associated with these three messages are the major parameters of OpenFlow performance measurement for the C2D and D2C parts. In the OFD part, performance parameters can be categorized into two classes—those related to packet processing and those related to the OpenFlow mechanism. The former category consists of table lookup, table pipeline, and time of action set execution. The latter category is composed of the timeout of flow entry and the size of the flow table.

B. Related Work

Some testing solutions with a focus on TD have been developed. Bianco *et al.* [10] and Emmerich *et al.* [11] proposed the method for measuring forwarding throughput and the packet latency of switches in underloaded and overloaded situations. Gelberger [12] proposed the cost measurement on throughput, latency, and jitter for switching and routing with OpenFlow. Jarschel *et al.* [17] presented a model based on forwarding speed and blocking probability to estimate packet sojourn time.

Recently, Spirent presented guidelines for testing OpenFlow switch performance [8] focusing on general testing concepts, and providing suggestions for each test cases. Rotsos *et al.* in OFLOPS [9] proposed two major test cases for measuring the latency of flow-mod and the execution time of an action set. Although the latency can be evaluated from OpenFlow barrier messages, the measurements are not accurate when a deviation may occur for two major reasons. The first is that barrier messages require extra time to be processed between the controller and the switch. The second is that the process of barrier messages in the OpenFlow switch may not be implemented correctly. The methods that were proposed by OFLOPS [9] solve these problems by using traffic to measure the setup time of multiflow entries. More recently, Rotsos *et al.* in OFLOPS proposed OFLOPS-Turbo [23] to evaluate the next-generation OpenFlow switch up to 10 Gb/s. However, the system does not address any method for detailed switch testing. Mafioletti *et al.* [20] proposed a tool capable of measuring accurate network device latency on a general purpose x86 hardware. With the design based on the Linux container technology, the system can support 100% time-stamping up to packet rate of 800 kp/s. He *et al.* [21] presented the measurement results of control plane latency on several commercial OpenFlow switches. Surprisingly, some high latency results are observed due to dependencies of rule complexity, rule priority, table priority, and limited bus bandwidth between ASIC and CPU. [22].

Since the latency of flow-mod may vary with the testing conditions, Handfield *et al.* [19] identified the factors that have the greatest impact on timing measurement, including the priority, the size of the flow table, and the operation of the flow-mod. Bianco *et al.* [10] and Tanyingyong *et al.* [24] estimated the effects of lookup time on the hash and linear flow table. Bianco *et al.* [10] and Tanyingyong *et al.*[24] showed that the hash flow table outperforms the linear flow table. With respect to the measurement of processing time, Bianco *et al.* [10] conducted experiments on hash and linear flows with various table sizes. The results are constant and independent of the table size for both types of table. The developed high-fidelity switch models [14] have proposed a model to emulate hardware switches, including the flow-mod rate. However, no detail methods has been provided. Lazaris *et al.* proposed Tango [25], a framework that is capable of revealing the SDN switch properties. Tango can generate a sequence of OpenFlow flow-mod commands and traffic patterns. By observing the outcomes of the switch under test, the capability, availability, and feature of the switch can be obtained. Instead of relying on the switch to provide the feature report via the OpenFlow protocol, the proposed active probing can provide more accurate results. By knowing the accurate features such as the TCAM size, caching policy, delay and rule processing priority, the application programmer can better optimize the SDN application achieving higher performance.

At the SDN controller side, R. Sherwood and K. Yap presented the *Cbench* [26] which became the*de facto* SDN controller benchmark standard. Tootoonchian *et al.* [27] designed a series of flow-based benchmark tools in *Cbench*, which is capable of emulating any number of OpenFlow switches to measure different performance aspects of the OpenFlow controller. Jarschel *et al.* [28] proposed another flexible and granular OpenFlow controller performance analysis system. Sieber *et al.* [29] presented the *hvbench* to address the issues of SDN hypervisor benchmarking. Based on the Linux container technology, each instance of *hvbench* node consists of multiple NOS and dataplane nodes. Therefore, the framework can provide the performance characteristics of the hypervisor under difference load scenarios. As shown in Table II, the relevant literature does not cover all of the performance parameters that are required to evaluate the OpenFlow switch under test. Therefore, this paper, proposes methods for measuring all of the major performance

TABLE II
RELATED WORKS

Type	Parameter	Spirent [8]	OFLOPS [9]	Bianco [10]	Handfield [19]	Mafioletti [20]	He [21], [22]	Proposed
D2C	Packet-in Rate	Concept	n/a	n/a	n/a	n/a	Timestamp	Buffer Size
C2D	Packet-out Rate	Concept	n/a	n/a	n/a	n/a	Timestamp	Buffer Size
	Latency of Flow-mod	n/a	Traffic Validation	n/a	Impact Factors	n/a	Insertion Modification Deletion	n/a
OFD	Table Lookup	Concept	n/a	Constant	n/a	n/a	n/a	n/a
	Table Pipeline	Concept	n/a	n/a	n/a	n/a	n/a	Busy Ratio
	Execution Time of Action Set	n/a	End-to-End	n/a	n/a	n/a	n/a	Action Time
	The Timeout of Flow Entry	Concept	n/a	n/a	n/a	n/a	n/a	Accuracy

TABLE III
TABLE OF NOTATIONS

Category	Notation	Description
Entity	c	The controller.
	dut	The switch under test.
	N	The number of tables for the switch under test.
	$H = \{h_n \mid n \geq 2\}$	The set of hosts.
	$CAP = \{cap_c, cap_n \mid n \geq 2\}$	The set of link capacities. The cap_c/cap_n is link capacity between c/h_n and dut.
C2D	$CD = POR$	The parameter for C2D. POR means the throughput of packet-out operation in dut.
D2C	$DC = PIR$	The parameter for D2C. PIR means the throughput of packet-in operation in dut.
OFD	$T_{\mathrm{action-set}}$	The time of action set execution in dut.
	$T_{\mathrm{table-pipeline}}$	The latency between two tables in the pipeline of dut
	buf	The size of buffer in the switch under test.
	Acc_{timeout}	The accuracy of timeout in the switch under test.
	P	The efficiency of table pipeline in the switch under test.
	$D_{\mathrm{openflow}} = \{T_{\mathrm{action-set}}, T_{\mathrm{table-pipeline}}, buf, Acc_{\mathrm{timeout}}, P\}$	The set of parameters for the OpenFlow dataplane, OFD.
Process	$TB = \{\text{table}_i \mid 0 \geq i < N\}$	The set of flow tables in the switch under test.
	$F = \{\text{flow}_{i,j} \mid 0 \geq i < N, j > 0\}$	The set of flow entries. $Flow_{i,j}$ mean the flow entry in table$_i$.
	TFC $= \{pkt_z \mid z > 0\}$	The traffic which is sent from src to dst. $src \in H, dst \in H$.
	$T_{\mathrm{table\_lookup}}$	The time of looking up the flow table in dut.
	$T_{\mathrm{apply\_action}}$	The time of executing Apply-Action in dut.
	$T_{\mathrm{idle-timeout}}, T_{\mathrm{hard-timeout}}$	The timeout value for idle/hard timeout flow entry.
	$T_{\mathrm{idle-expired}}, T_{\mathrm{hard-expired}}$	The arrival time of flow-removed message for idle/hard timeout flow entry at c.
	$T_{\mathrm{idle-duration}}, T_{\mathrm{hard-duration}}$	The duration of flow-removed message for idle/hard timeout flow entry.

parameters, except for the latency of flow-mod and the size of the flow table.

III. PROBLEM STATEMENT

Fig. 2 presents the major performance parameters measured of the device under test, denoted as dut including action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy. The pipeline process contains the time and performance of table pipeline ($T_{\mathrm{table-pipeline}}$, P), and the time of action set execution $T_{\mathrm{action-set}}$ in the dut. Table III shows the notations used in this proposed suite. Given the switch under test dut, the number of tables in the dut is denoted as N. The OpenFlow controller, the hosts, and the link capacities are denoted as c, H, and CAP. The set of flow entries F determines the operation of the entire pipeline. Each flow entry, denoted by $flow_{i,j}$, has two timers of hard-timeout and idle-timeout. The accuracy of the timeout, denoted as Acc_{timeout}, evaluate whether each timer expired correctly or not. With the combinations of F and traffic, denoted as TFC, the dut is able to generate the packet-in messages to the OpenFlow controller c and process the packet-out messages sent from the controller. The rate of packet-in and packet-out for a given dut is determined by the parameters of PIR and POR. The buffer size of dut is denoted as buf. If there is room left in the buffer, dut stores the TFC in the buffer and raises the packet-in operation with an identity as a new flow arrives. The controller c then generates the packet-out message with the corresponding identity to the dut to forward the new flow.

The performance parameters consist of three categories, which are the C2D parameters CD, the D2C parameters DC, and the OFD parameters D_{openflow}. The parameters of flow entries F in dut and traffic TFC generated by H are mutual related.

The testing environment is created based on the given entities shown in Table III. The flow entries and traffic are determined based on the values of F and TFC to evaluate the performance parameters of CD, DC, and D_{openflow}. The objective of this paper is to assure the accuracy of each measurement result of CD, DC, and D_{openflow}.

IV. METHODOLOGY

Table IV presents the measured performance parameters in the proposed five test cases. These cases fall into three categories, which are mirror-in-processing, masked entry, and

TABLE IV
LIST OF FIVE TEST CASES PROPOSED

Category	Test Case	Method	Output
Mirror-in-Processing	Action Set time	Mirror-first-then-action	$T_{\text{action}-\text{set}}$
	Pipeline Time	Mirror-first-then-pipeline	$T_{\text{table}-\text{pipeline}}$
Calculated Traffic	Buffer Size	Burst-until-loss	buf, PIR, POR
	Pipeline Efficiency	Back-to-back-traffic	P
Masked Entry	Timeout Accuracy	Idle-timeout-derived-by-hard-timeout	Acc_{timeout}

calculated traffic. The measurement of packet-in and packet-out rates are also accompanied by the buffer size test case. Two test cases are presented to obtain the time to evaluate the pipeline processing performance. In addition to the end-to-end latency measurement proposed by OFLOPS [9], the proposed test case for measuring the execution time of the action set is provided. Finally, this paper proposes a method to verify the accuracy of an idle and hard timeout for the flow entry.

A. Mirror-in-Process

In OpenFlow performance testing, it is difficult to measure the internal processing latency directly. However, by leveraging the mirror operation with Apply-Actions instruction embedded, the packets can be duplicated before some OpenFlow operations are performed for measurement and comparison.

1) Mirror-First-Then-Action: In this test case, the execution time of an action set in an OpenFlow switch is measured. The OpenFlow protocol include many kinds of actions, which comprise forwarding, packet modification, and the tagging operation. Packet modification may increase the operation time in some switches. Therefore, the processing latency of an action is used as a major performance benchmark.

In a normal case, the time of action, denoted as $T_{\text{action}-\text{set}}$, can be calculated from measured end-to-end latency. Since many variables influence the testing result, we propose mirroring to obtain the start time of an action set to improve measurement accuracy. As displayed in Fig. 3, pkt_z denotes the mirrored packet. The value of α is the arrival time of pkt_z and β is the arrival time of pkt_z‘. The execution time for the action set is derived as

$$T_{\text{action}-\text{set}} = \beta - \alpha. \tag{1}$$

2) Mirror-First-Then-Pipeline: In this case, the time of table pipeline in OpenFlow switch is measured. The mirroring method can be applied to this test case as well, because the instructions of table pipeline are performed after Apply-Actions. As presented in Fig. 4, the pipeline stage is assumed to comprise of two tables—$table_0$ and $table_1$. The flow entries in these two tables are mostly the same, except the flow entry in $table_0$ has GoTo instruction. The time T_1 includes the time for table lookup and Apply-Actions operations in $table_0$. The time T_2 includes the time $T_{\text{table}-\text{pipeline}}$, the time for table lookup, and the time

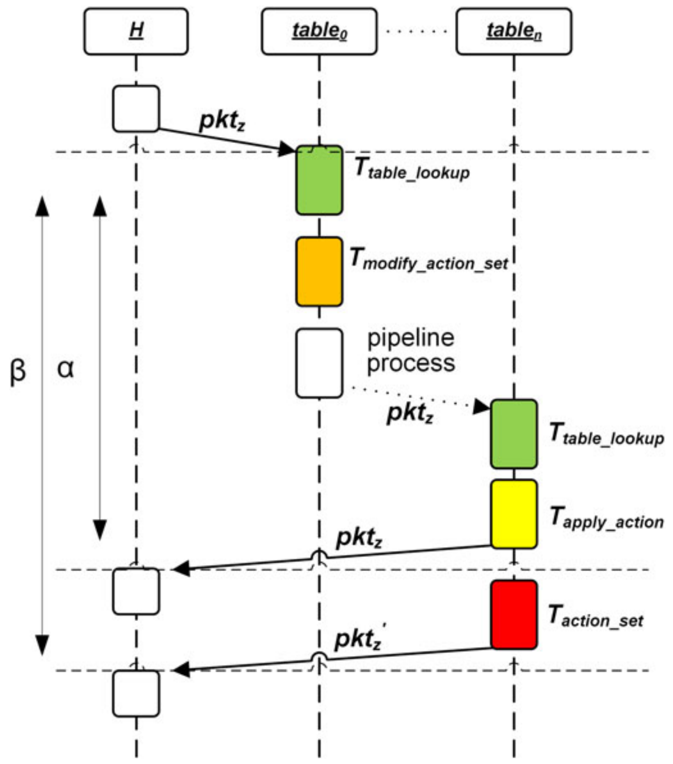

Fig. 3. Process of the proposed mirror-first-then-action method for action time measurement. The execution time for the action set can be estimated by $T_{\text{action}-\text{set}} = \beta - \alpha$.

for Apply-Actions operations in $table_1$. Since flow entries in $table_0$ and $table_1$ have the same match fields and actions, the processing latency of table lookup and Apply-Action, denoted as $T_{\text{table\_lookup}}$ and $T_{\text{apply\_action}}$ respectively, are the same in $table_0$ and $table_1$. From the round-trip time (RTT), the time of the table pipeline can be derived as

$$T_{\text{table}-\text{pipeline}} = T_2 - T_1 + \text{RTT}/2 \tag{2}$$

where the T_1 is

$$T_1 = T_{\text{table\_lookup}} + T_{\text{apply\_action}} + \text{RTT} \tag{3}$$

and the T_2 is

$$\begin{aligned} T_2 = \; & T_{\text{table}-\text{pipelien}} + T_{\text{table\_lookup}} \\ & + T_{\text{apply\_action}} + \text{RTT}/2. \end{aligned} \tag{4}$$

The RTT can be derived by $\text{RTT} = (pkt_z/cap_n) * 2$.

B. Calculated Traffic

1) Burst-Until-Loss: In this test case, the buffer size, the throughput of packet-in and the throughput of packet-out are measured. The measurement of these three parameters is performed in one case because the packet-in and packet-out messages may depend on the size of the buffer in the dut .

As shown in Fig. 5, the dut has the flow entry that matches the TFC and sends the packet-in message to the controller c. Controller c stores all packet-in messages and generates the expected packet-out messages after the traffic that is generated

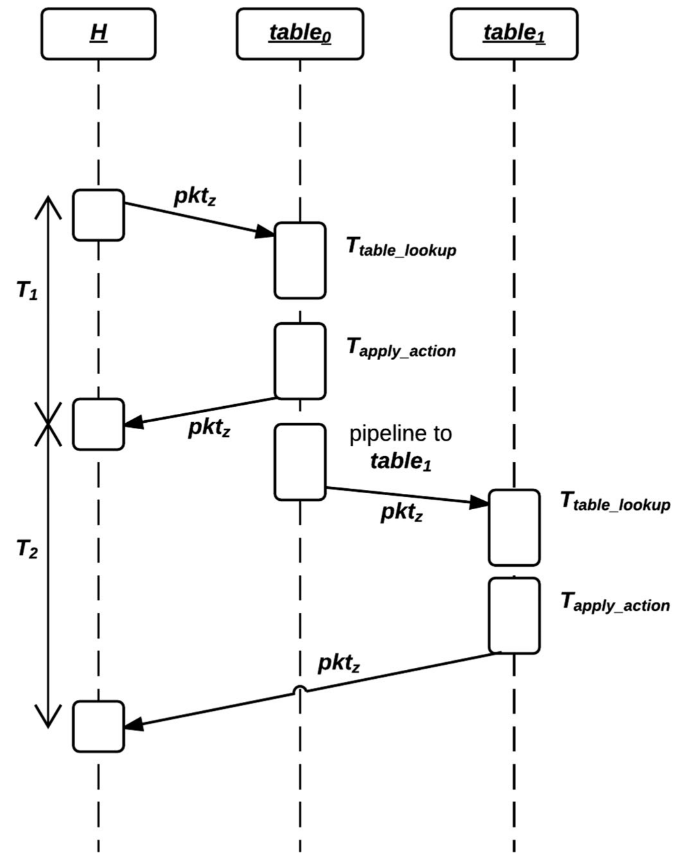

Fig. 4. Process of proposed mirror-first-then-pipeline method for the latency measurement between two tables.

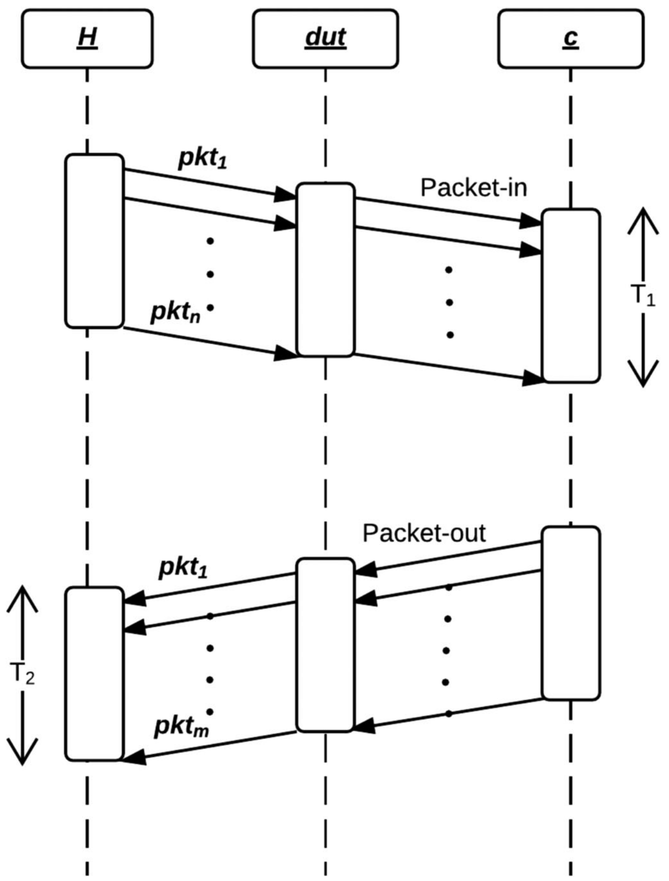

Fig. 5. Process of proposed burst-until-loss method for buffer size estimation.

from host H is done. It is assumed that there is no packet loss occurred at c and all packet-in messages consume the entire buffer space in the dut. Under the condition $m < n$, packet loss occurs at the dut. Therefore, the buffer size, denoted as buf can be expressed as

$$buf = m * pkt\_size. \tag{5}$$

The throughput of packet-in (PIR) and the throughput of packet-out (POR) can be expressed by the formulas as

$$PIR = \text{the number of Packet-in}/T_1 \tag{6}$$

and

$$POR = m/T_2. \tag{7}$$

2) Back-to-Back-Traffic: In this test case, the efficiency, also known as the busy ratio of the pipeline in dut is measured. As displayed in Fig. 6, the latency of TFC includes the total processing time for $n+1$ tables. In each table, packets are matched with the flow entry and sent to the next table in a pipeline fashion. The busy ratio of the pkt processing time in each table can be derived from the packet processing time and idle times. The processing time of pkt and the idle time are denoted as T_{pkt} and T_{bubble}, respectively. The busy ratio is defined by the formula $P = T_{\text{pkt}}/(T_{\text{bubble}} + T_{\text{pkt}}) \times 100\%$, where T_{pkt} is the processing time of each packet in each table. Owing to the difficulty of measuring the T_{pkt} in black-box testing, uniform processing time of pkt in all tables are assumed for the performance evaluation. Therefore, T_{pkt} can be estimated as

$$T_{\text{pkt}} \approx T_{\text{init}}/(n+1) \tag{8}$$

where T_{init} represents the processing time of the first packet.

The time $T_{\text{bubble}} + T_{\text{pkt}}$ is the total time of processing pkt and the idle time for a single table. In this measurement, the total number of packets $m+1$ and the total duration T are known. Therefore, assuming m packets are sent after the first packet, the mean time $T_{\text{bubble}} + T_{\text{pkt}}$ can be derived as

$$T_{\text{bubble}} + T_{\text{pkt}} \approx (T - T_{\text{init}})/n. \tag{9}$$

T_{init} and T are the processing time for the first packet, and the total processing time for $m+1$ packets, respectively. The time of sending pkt_z is denote as α_z and the arrival time of $pkt_z$is denoted as β_z. T_{init} and the T are given by the $T_{\text{init}} = \beta_1 - \alpha_1$ and $T = \beta_{m+1} - \alpha_1$, respectively.

C. Masked Entry

Based on the specification of OpenFlow, the flow table can have multiple entries with the same match fields but with different priorities.

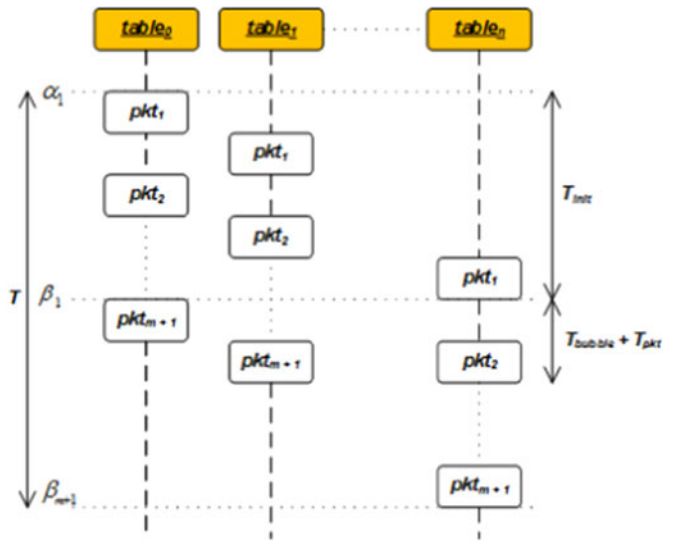

Fig. 6. Process of proposed back-to-back-traffic method for pipeline efficiency measurement. For each packet in a table, the busy processing time and idle time are represented by the rectangle and dash line, respectively.

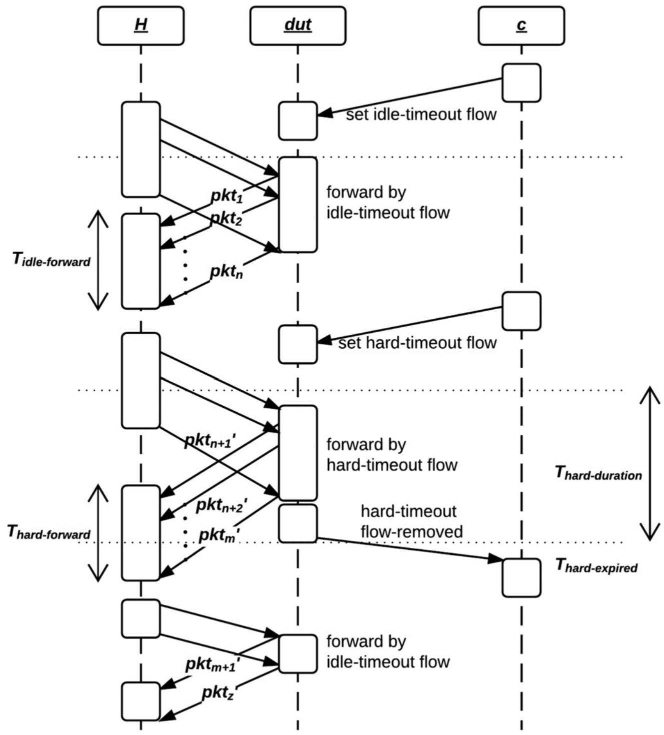

Fig. 7. Idle-timeout derived from hard-timeout: The hard-timeout flow entry expired before the one of idle-timeout indicating skewness of the idle-timeout timer.

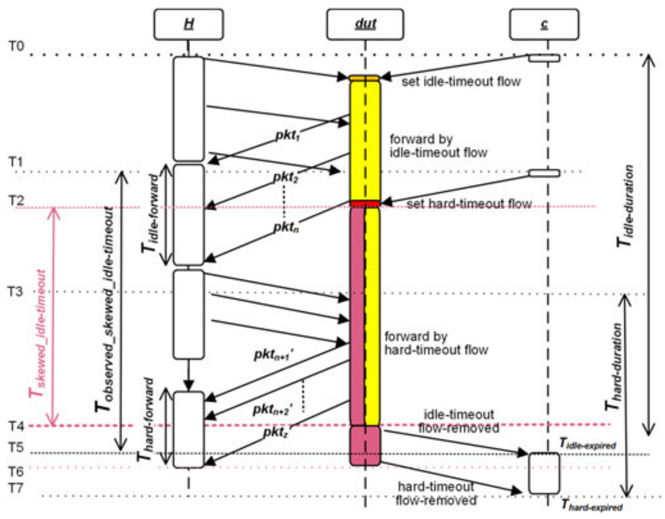

Fig. 8. Idle-timeout derived from hard-timeout: The skewed idle-timeout (T2∼T4) can be observed at the controller (T1–T5). This value can be derived by subtracting the $T_{\text{idle-duration}}$ and $T_{\text{idle-forward}}$ assuming a neglectable processing and propagation delay.

1) Idle-Timeout-Derived-by-Hard-Timeout: In this case, the accuracy of timeout for a given flow entry is measured. The measured timeout includes hard-timeout and idle-timeout. The accuracy of the measurement of hard-timeout can be evaluated simply from the packet that is injected into the DUT. However, the deviation of the accuracy of idle-timeout measurement may existed owing to the interarrival time of the traffic when the flow entry is matched.

To evaluate the accuracy of idle-timeout, use of the masked entry is proposed to avoid reseting of the timer at the wrong time by the arriving packet. As shown in Fig. 7, two flow entries are added to the dut. The first is the idle-timeout flow entry and the second is the hard-timeout flow entry, which is the masked entry with the higher priority. Both flow entries are matched with the TFC, and the time of idle-timeout ($T_{\text{idle-timeout}}$) equals the time of hard-timeout ($T_{\text{hard-timeout}}$). When the idle-timeout flow entry is added to the dut, the timer of idle-timeout remains at zero as the packets are forwarded based on the matched entry. After the idle-timeout flow entry is set, the hard-timeout flow entry is added to the dut. At this time, the hard-timeout flow entry takes over the idle-timeout flow entry because it has higher priority. The timers for both of the flow entries begin counting from zero as they are synchronized. Figs. 7 and 8 illustrate two situations. In Fig. 7, the hard-timeout flow entry expires before the idle-timeout flow entry, so the timer for idle-timeout is skewed. To evaluate this skewness, $T_{\text{hard-timeout}}$ is increased and the hard-timeout flow entry is set again.

The aboveprocess is repeated until the idle-timeout flow entry has expired early as illustrated in Fig. 8. The accuracy of idle-timeout can be expressed as

$$\text{Acc}_{\text{idle-timeout}} = (T_{\text{idle-duration}} - T_{\text{idle-forward}} - T_{\text{idle-timeout}})/T_{\text{idle-timeout}} \times 100\%. \quad (10)$$

The terms $T_{\text{idle-duration}}$ and $T_{\text{idle-forward}}$ represent the alive and active times for idle-timeout flow, respectively. The arrival time of pkt_n is denoted as α_n. The value of $T_{\text{idle-forward}}$ can be derived by using the formula of

$$T_{\text{idle-forward}} = \alpha_n - \alpha_1. \quad (11)$$

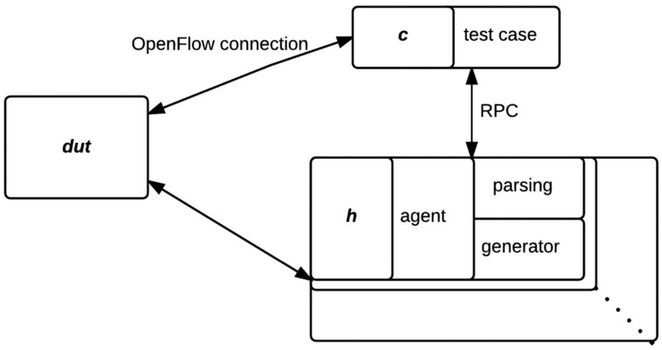

Fig. 9. Block diagram of OFBench architecture.

V. System Implementation

The proposed OFBench is an automatic testing suite which is integrated with a Ryu controller[1] and an Ostinato traffic generator.[2] The architecture of OFBench, displayed in Fig. 9, is a controller-agent architecture. Controlled by the controller, agents can generate traffic, capture packets, and parse packets. The test cases of OFBench are written in scripts for basic OpenFlow operations and the analysis of test results.

A. Test Setup

Five test cases are developed using test scripts in the OFBench suite. The available configurations of the test cases consist of frame size, transmission rate, and some specific values. The available frame sizes are 64, 128, 256, 512, and 1024 bytes, and the available transmission rates range from 64 kb/s to 1Gb/s. In our experiment, UDP traffic is used in all test cases. Experiments are conducted with three physical desktop computers. The Open vSwitch is setup based on the Ubuntu 14.04 OS on a commodity Intel PC with two NICs without extra hardware accelerator. The host (*h*) and controller (*c*) are constructed based on the Intel Core-i7 processors running at 3.3 GHz with Ubuntu 14.04 of kernel 4.2.0-27.

B. Experimental Results

In this section, five performance metrics—action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy—are evaluated using the five proposed test cases. The environment that is presented in Fig. 9 is used to test the four DUTs listed in Table V. The statistical results are collected with five rounds of testing.

1) Action Time: The relationship between action time and the combination of actions is sought. The tested actions comprise sets of packet header modifications from layer two to four. Fig. 10 presents the results concerning action time with different number of actions of switches under test. The modifications of packets are based on the combinations of Ethernet (eth), IP, and UDP protocols. In each testing scenario, traffic of minimum-sized frame is generated at the rate of 1000 packets per second (pps) for a total of up to 150 000 packets.

With respect to Fig. 10, the software switch has the lowest average action time of 6 μs. The action time measured ranges from 3 to 23 μs. The standard deviation of the measured action time is shown in Fig. 11. The standard deviation grows with the increase of frame rate and number of actions. We further conducted a test case based on a white-box environment to verify the test result. The kernel trace tool (ftrace) is used to record the execution time of the *set_ip_addr* function for Open vSwitch. The white-box-based measurement on the action time is approximately 0.6 μs.

The average action time for hardware switches is at least 100 μs. According to the specifications in Table V, the CPU in the Open vSwitch has the highest clock rate compared to those of the hardware switches. The measured action time that is presented in Fig. 10 is correlated with the CPU clock frequency. The results for the hardware switch SW4 are unavailable because the mirror and action do not execute properly.

2) Pipeline Time: In the pipeline time test case, the results are unavailable for switches SW1 and SW2 because Apply-Action is not working properly. Those switches can not mirror packets between tables of the pipeline process. Therefore, only results for Open vSwitch, SW3, and SW4 with various frame sizes from 64 to 1024 bytes are available. The measured pipeline time is in the range of 1.1 to 2 μs positively correlated with frame size.

3) Buffer Size: In the buffer size test case, the packet-in rate, packet-out rate, and buffer size of the switch under test are measured in a single test. The transmission rate is fixed at 1Gb/s with various frame sizes for the measurement. Fig. 12 presents the packet-in rates. The results that are collected at the beginning of the test are discarded, mainly because the software-based traffic generator takes seconds to stabilize the testing traffic. According to the results shown in Fig. 12, the rate of packet-in for Open vSwitch is approximately three times lower than that of the hardware switch SW2, indicating that the software switch poorly handles the small frame size in packet-in operation.

With respect to the packet-out rate, both switches of SW1 and SW2 poorly handle small frames with the size of 64 bytes. However, the measurements results in Open vSwitch are different. The packet-out operation works well with the frame size of 64 bytes, and the throughput of packet-out for Open vSwitch is better than those observed in other switches under test.

Table VI presents the measured results of buffer sizes of switches under test. Some error messages are generated by the DUTs during testing. The errors, identified as “buffer unknown” for packet-out messages result in the failure of the packet forwarding in the DUT for each corresponding packet-out received. Sometimes, the number of error messages exceed the number of packet-out messages that are sent from the controller. This test results reveal that packet loss has occurred at the DUTs. Therefore, the buffer size of the DUT is not presented as errors occurred. In Table VI, only buffer sizes that are tested with a frame size of 64-byte are shown. For other frame sizes, the buffer sizes of the switches can not be estimated due to the errors stated above. We suspect that the anomalous behavior of DUTs

[1][Online]. Available: Ryu, version 4.0, https://osrg.github.io/ryu/
[2][Online]. Available: Ostinato, version 0.7.1, http://ostinato.org/

TABLE V
SPECIFICATIONS OF HARDWARE AND SOFTWARE SWITCHES

Switch	CPU	Core	Clock rate	Memory	Buffer	OS version
Pica8 P-3290	MPC8541	1	1 GHz	512 MB	4 MB	v2.6.1
Pica8 P-3297	P2020	2	1.33 GHz	2 GB	4 MB	v2.6.1
Edge-corE AS4610-30T	ARM Cortex A9	2	1 GHz	2 GB	N/A	v2.6.4
Centec V350	e500v2	1	533 MHz	2 GB	N/A	v3.1(11), 1.alpha
Open vSwitch	Intel Core i3	2	3.1GHz	8 GB	N/A	Ubuntu 14.04, 3.13.0-24

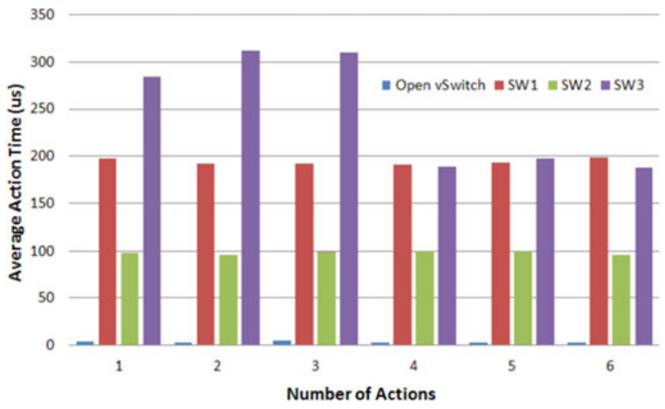

Fig. 10. Measured average action time for switches under test.

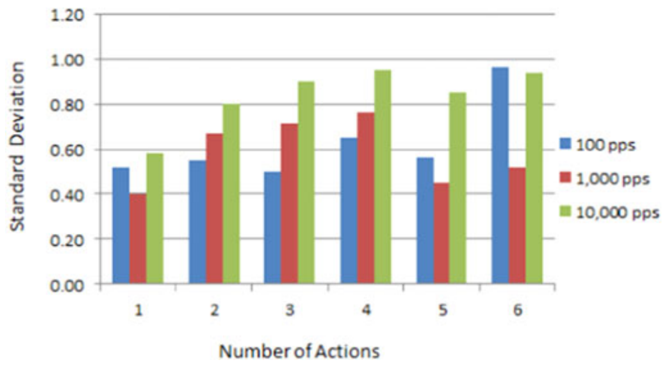

Fig. 11. Standard deviation of action time measured for the Open vSwitch. The tests are conducted under various traffic loads of 100, 1 k, and 10 k packet/s with 150 k number of frames.

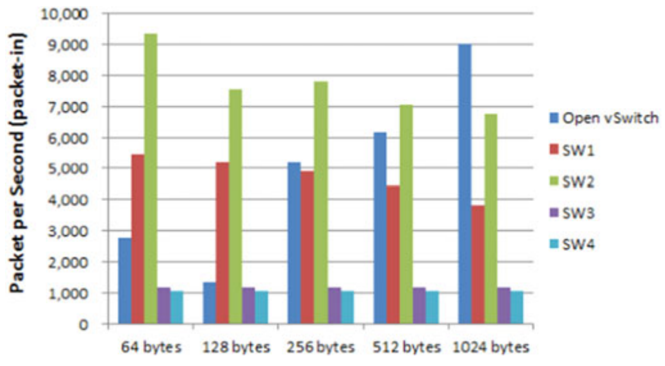

Fig. 12. Measured packet-in rates for switches under test during the test of buffer size with generated traffic rate of 1 Gb/s.

TABLE VI
BUFFER SIZE ESTIMATION WITH 64-byte FRAME SIZE UNDER 1-Gb/s TRAFFIC

Open vSwitch	SW1	SW2	SW3	SW4
13 504	8 256	16 192	32 768	342 464

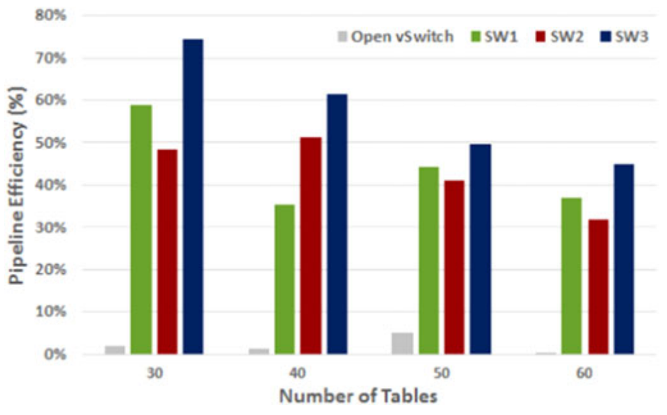

Fig. 13. Estimated efficiency of table pipeline with different number of tables (30– 60) for switches under test. The low percentage of efficiency indicates long wait time in the pipeline process.

arises due to the buffer overflow issue of CVE-2016-2074 [30] in Open vSwitch implementation.

4) Pipeline Efficiency: With respect to the measurement of efficiency for table pipeline operation, the test case is conducted to go through 60 tables with transmission rates of 1024 Mb/s. As shown in Fig. 13, the software switch has poor efficiency on the operation of traversing multiple tables. For hardware switches, switch SW3 achieves a busy ratio above 70% and the pipeline efficiency is inversely proportional to the number of tables. This indicates the software switch has longer idle time compared to those of the hardware switches on multitable pipelining. The result for switch SW4 is not available due to the switch only has a single table. During the test, error messages were observed at the Open vSwitch with a large amount of traffic in the test. The sent and received packets were mismatched when the Open vSwitch is operated under heavy loading, making the busy ratio calculation unavailable.

5) Timeout Accuracy: In the evaluation of timeout accuracy, the idle-timeout was set to 5 s and the hard-timeout was varied from 3 to 7 s. The traffic is generated at a rate of 10 K pps with a frame size of 64-byte. Table VII presents the results thus obtained. The deviation in the idle-timeout of hardware

TABLE VII
RESULT OF TIMEOUT ACCURACY FOR SWITCHES UNDER TEST

Switch	$Acc_{\mathrm{idle-timeout}}$	$Acc_{\mathrm{hard-timeout}}$
Open vSwitch	2%	0%
SW1	12.47%	4.7%
SW2	11.39%	3.3%
SW3	4.7%	0%
SW4	17.64%	1.66%

The idle-timeout is set to 5 s and the hard-timeout was varied from 3 to 7 s.

switches is approximately less than 20%. A larger deviation for the idle-timeout is observed in switch SW4 because the timer for idle-timeout can not be reset properly when the idle-timeout flow entry is matched.

C. Processing Overhead

Currently in the prototype development phase, the system conducts post analysis on pcap files produced during each test. The size of these pcap files is proportional to the duration of the testing. Therefore, some performance overheads are imposed in the process of file storage while generating and receiving the traffic frames. The variations might occurred at the host side due to the performance of NICs, drivers, and CPUs. As far as the controller is concerned, only test cases of "*burst-until-loss*" and "*timeout accuracy*" might be affected due to the insufficient controller performance.

For the test case of evaluating idle-timeout accuracy, the mechanism is based on the setting of two flow entries of different priority. It takes time for the DUTs to process the incoming packet for matching the new rule while the other ages out causing the timer not reset properly. Therefore, variation might occurred at this gray area. Moreover, as the calculation is based on the value of hard-timeout, the accuracy of hard-timeout operation in the DUTs might affect the variations of the idle-timeout measurement.

VI. Conclusion and Future work

This paper presents an automatic test suite that is based on the controller-agent architecture. It can control remote agents to generate and analyze networking traffic. The suite implementation is focused on the testing of OpenFlow switch performance with respect to the following aspects; control-plane-trigger-data-plane (C2D), data-plane-trigger-control-plane (D2C), and OFD. Five test cases are considered to evaluate five performance metrics, which are action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy. These five metrics can be used to evaluate the performance of a switch under test. According to the results of testing with minimum-sized frame herein, the hardware switches can generate packet-in messages at rates from 1000 to 9000 messages per second. The rate is approximately three times higher than that generated by the software switch. With respect to timeout accuracy, the hardware switches exhibit a deviation less than 20% in idle timeout. The larger idle timeout deviation of hardware switches may cause an error in applications that are based on the timer of idle-timeout. Measurements of the pipeline efficiency revealed that, hardware switches had a better pipeline efficiency. When pipeline was enabled and tested under a 1Gb/s traffic loading, the hardware switches reached a pipeline efficiency approximately above 50% with 30 tables.

Generally, the hardware switch is able to better handle new flow than is the software switch. Some interesting issues and unknown behaviors were observed during the testing for the OpenFlow switches. For example, some switches crashed when the volume of burst packet-in traffic was high and, the Apply-Action instructions may not be well implemented in some switched. The timer of idle-timeout is not reset properly when the flow entry is matched.

In the experiments herein, performance factors that were affected by the loading of the switches were excluded from consideration, possibly causing some variance in the obtained testing results. We plan to consider these factors and discuss some advanced issues related to our experiments in the near future. As part of the ACTS development, the tests can be easily deployed in the virtual environment by redirecting the outcomes of the traffic generator and the responses from the DUT to a virtual host fulfilling the performance measurement purpose. Since the original system design is based on a controller-agent architecture with high-level script language, test cases can be modified to generate any given combinations of OpenFlow messages addressing not only the needs of performance measurement but also the issues of switch diversity. In the near future, the mechanisms of on-demand and real-time analysis will be implemented. Pcap filters will be added to avoid the unnecessary frames recorded in the pcap file. The calculation of the queuing time will be included in the pipeline time, making the measurement of pipeline performance more accurate.

Acknowledgment

This work would not be possible if without the strong support from Network Benchmarking Lab (NBL, www.nbl.org.tw), which is the 5th ONF authorized test lab. In particular, the authors would like to thank Dr. E.-C. Liou, Y.-T. Chiang, and Executive Director C.-H. Hsu, for their direct contribution and support in designing the OFBench related test methodologies and tools.

References

[1] Open Network Foundation, 2017. [Online]. Available: https://www.opennetworking.org
[2] The Open Networking Foundation, *OpenFlow Switch Specification V1.3.0*, 2012. [Online]. Available: https://www.opennetworking.org/
[3] S. Bradner and J. McQuaid, "Benchmarking methodology for network interconnect devices," RFC 2544 (Informational), Mar. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2544.txt
[4] D. Raumer, S. Gallenmüller, F. Wohlfart, P. Emmerich, P. Werneck, and G. Carle, "Revisiting benchmarking methodology for interconnect devices," in *Proc. 2016 Appl. Netw. Res. Workshop*, 2016, pp. 55–61.
[5] OFTest, 2014. [Online]. Available: http://www.projectfloodlight.org/oftest/
[6] Ryu Certification, 2014. [Online]. Available: https://osrg.github.io/ryu/certification.html
[7] M. Kuzniar, P. Peresini, M. Canini, D. Venzano, and D. Kostic, "A soft way for openflow switch interoperability testing," in *Proc. 8th Int. Conf. Emerging Netw. Exp. Technol.*, 2012, pp. 265–276.

[8] Spirent, "Openflow performance testing white paper," 2015. [Online]. Available: https://www.spirent.com/

[9] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An open framework for OpenFlow switch evaluation," *Proc. 13th Int. Conf. Passive Active Measurement*, 2012, pp. 85–95.

[10] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow switching: Data plane performance," in *Proc. IEEE Int. Conf. Commun.*, 2010, pp. 1–5.

[11] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance characteristics of virtual switching," in *Proc. 2014 IEEE 3rd Int. Conf. Cloud Netw.*, 2014, pp. 120–125.

[12] A. Gelberger, N. Yemini, and R. Giladi, "Performance analysis of software-defined networking (SDN)," in *Proc. IEEE Comput. Soc. Annu. Int. Symp. Model., Anal., Simul. Comput. Telecommun. Syst.*, 2013, pp. 389–393.

[13] D. Jin, D. M. Nicol, and M. Caesar, "Efficient gigabit ethernet switch models for large-scale simulation," in *Proc. 2010 IEEE Workshop Princ. Adv. Distrib. Simul.*, 2010, pp. 122–131.

[14] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 43–48.

[15] Z. Shang and K. Wolter, "Delay evaluation of openflow network based on queueing model," *CoRR*, vol. abs/1608.06491, 2016. [Online]. Available: http://arxiv.org/abs/1608.06491

[16] T. Sato, S. Ata, I. Oka, and Y. Sato, "Abstract model of SDN architectures enabling comprehensive performance comparisons," in *Proc. 2015 11th Int. Conf. Netw. Service Manage.*, Nov. 2015, pp. 99–107.

[17] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *Proc. 23rd Int. Teletraffic Congress*, 2011, pp. 1–7.

[18] C.-H. Hsu, "Automation control test system," Network Benchmarking Lab, 2016. [Online]. Available: https://www.nbl.org.tw/services_zh-tw_31_35.html

[19] M. Kuźniar, P. Perešíni, and D. Kostić, "What you need to know about SDN flow tables," *Passive and Active Measurement. PAM 2015.* (Lecture Notes in Computer Science), vol. 8995, J. Mirkovic and Y. Liu, Eds., Cham, Switzerland: Springer, 2015, pp. 347–359.

[20] D. R. Mafioletti, A. B. Liberatto, R. d. S. Villaça, C. K. Dominicini, M. Martinello, and M. R. N. Ribeiro, "Latency measurement as a virtualized network function using metherxis," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 4, pp. 14–16, Dec. 2016.

[21] K. He *et al.*, "Latency in software defined networks: Measurements and mitigation techniques," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 435–436, Jun. 2015.

[22] K. He *et al.*, "Measuring control plane latency in SDN-enabled switches," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, pp. 25:1–25:6.

[23] C. Rotsos, G. Antichi, M. Bruyere, P. Owezarski, and A. W. Moore, "OFLOPS-Turbo: Testing the next-generation OpenFlow switch," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5571–5576.

[24] V. Tanyingyong, M. Hidell, and P. Sjodin, "Improving PC-based OpenFlow switching performance," in *Proc. 2010 ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2010, pp. 8–9.

[25] A. Lazaris *et al.*, "Tango: Simplifying SDN control with automatic switch property inference, abstraction, and optimization," in *Proc. 10th ACM Int. Conf. Emerging Netw. Exp. Technol.*, 2014, pp. 199–212.

[26] K. Y. Robert Sherwood, *Cbench Controller Benchmark Tool*, 2011. [Online]. Available: https://github.com/mininet/oflops

[27] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. 2nd USENIX Conf. Hot Top. Manag. Internet, Cloud, Enterprise Netw. Serv.*, 2012, pp. 10–10.

[28] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A flexible openflow-controller benchmark," in *Proc. 2012 Eur. Workshop Softw. Defined Netw.*, Oct. 2012, pp. 48–53.

[29] C. Sieber, A. Blenk, A. Basta, and W. Kellerer, "hvbench: An open and scalable SDN network hypervisor benchmark," in *Proc. IEEE NetSoft Conf. Workshops*, Jun. 2016, pp. 403–406.

[30] CVE-2016-2074: MPLS Buffer Overflow Vulnerabilities in Open vSwitch, 2016. [Online]. Available: http://openvswitch.org/pipermail/announce/2016-March/000082.html

Ying-Dar Lin (F'13) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 1993.

He is a Distinguished Professor of computer science with National Chiao Tung University (NCTU), Hsinchu, Taiwan. He was a Visiting Scholar with Cisco Systems, San Jose, CA, USA, during 2007–2008. Since 2002, he has been the founder and the Director of the Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic and has been an approved test lab of the Open Networking Foundation since July 2014. He also cofounded L7 Networks, Inc., in 2002, which was later acquired by the D-Link Corporation. He is also an ONF Research Associate. He co-authored the textbook *Computer Networks: An Open Source Approach* (www.mhhe.com/lin) with R.-H. Hwang and F. Baker (McGraw-Hill, 2011). His research interests include network security and wireless communications. His work on multihop cellular was the first along this line and has been cited more than 750 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced.

Dr. Lin is an IEEE Distinguished Lecturer (2014–2017). He currently serves on the Editorial Boards of several IEEE journals and magazines.

Yu-Kuen Lai (S'03–M'06–SM'14) received the M.S. and Ph.D. degrees in electrical and computer engineering from North Carolina State University at Raleigh, NC, USA in 1997 and 2006, respectively.

He is an Associate Professor with the Electrical Engineering Department, Chung-Yuan Christian University (CYCU), Chung-Li, Taiwan. From 1997 to 2002, he was a Senior ASIC Design Engineer with Delta Networks, Inc., and the Applied Micro Circuit Corporation (AMCC) at Research Triangle Park, NC, USA. His research interests include network processor architecture, streaming data processing, network traffic analysis, FPGA systems design, and computer network security.

Dr. Lai served as the Electronic Communication Officer for the IEEE Taipei Section in 2015. He was the recipient of the CYCU Distinguished Teaching Award and CYCU Outstanding Teaching Award in 2011 and 2017, respectively.

Chen-You Wang received the M.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2016.

He is currently a Software Engineer with Proscend Communications, Inc., Hsin-Chu, Taiwan. His research interests include embedded system design, computer networking, and performance evaluation.

Yuan-Cheng Lai (A'05–M'13) received the Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1997.

In August 1998, he joined the faculty of the Department of Computer Science and Information Science, National Cheng Kung University, Tainan, Taiwan. In August 2001, he joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, where he has been a Professor since February 2008. His research interests include performance analysis, protocol design, wireless networks, and web-based applications.