Performance Evaluation Mechanisms for FlowMod Message Processing in OpenFlow Switches

Anh Nguyen-Ngoc*, Stanislav Lange*, Steffen Gebert*, Thomas Zinner*, Phuoc Tran-Gia*, Michael Jarschel§
*Institute of Computer Science, University of Würzburg, Germany.
Email: {anh.nguyen,stanislav.lange,steffen.gebert,zinner,trangia}@informatik.uni-wuerzburg.de
§ Nokia Bell Labs, Munich, Germany.
Email: michael.jarschel@nokia.com

***Abstract*—Network operators can benefit in terms of flexibility, cost, and vendor-independence when adopting the Software Defined Networking (SDN) paradigm. In many scenarios, the SDN controller orders the installation of new flow table entries in the switches it manages. Since such operations are handled in the slow path of the switches, the corresponding processing times constitute an important performance indicator for switches. This work focuses on a comparison of two mechanisms for evaluating the performance of OpenFlow switches with respect to the processing time of FlowMod messages. These mechanisms are characterized by different degrees of accuracy, cost, complexity, and the capability of performing measurements at run time.**

The first mechanism is based on the Spirent C1 dedicated testing platform, while the other uses a software module for the OpenDaylight controller. We assess their capabilities with respect to the abovementioned characteristics and quantify their accuracy by means of wiretaps that provide a ground truth regarding the measured processing times. By using three different switches in the experiments, it is possible to distinguish between hardware specific side-effects and general phenomena.

I. Introduction

The Software Defined Networking (SDN) paradigm changes several aspects regarding the operation and structure of today's networks. The key characteristics of the resulting architecture include the separation of control and data plane as well as a logically centralized control plane. This is achieved by moving control plane functions from the network devices to a dedicated controller software running on commercial off-the-shelf (COTS) hardware. Communication between this centralized control plane and the data plane takes place via the southbound API [1], an open interface which is implemented by protocols like OpenFlow [2].

Before moving to an SDN-based network deployment, operators need to make sure that the resulting network meets their particular requirements with respect to performance. This includes the data plane, i.e., packet forwarding, as well as the control plane. As a key performance indicator in the control plane, our work focuses on the processing times of FlowMod messages in an OpenFlow switch. Different mechanisms that can be used in the context of evaluating the FlowMod processing performance of switches are presented and compared with each other. Such mechanisms range from purely software-based approaches to sophisticated solutions which utilize hardware-based traffic generators and capture devices. Thus, they represent trade-offs in terms of cost, ease of use, and accuracy. In this context, the results obtained by means of wiretaps and dedicated capture devices are used as ground truth for accuracy assessment. The mechanisms analyzed in this work include a software-based approach module for the OpenDaylight controller as well as an approach based on the Spirent C1 testing platform.

In order to identify common phenomena as well as vendor-specific aberrations regarding switch behavior and performance, three different switch models from three different vendors are used as devices under test (DuT).

This work is structured as follows. An overview of related work is provided in Section II. Section III covers different types of FlowMod-related communication schemes defined in the OpenFlow specification as well as the testbed setup and the different approaches for evaluating OpenFlow switch performance used in this work. After discussing the results of the measurements in Section IV, Section V concludes the paper.

II. Related Work

This section provides an overview of related work regarding approaches for evaluating the performance of different aspects and components of an SDN architecture.

Possibilities for testing the network as a whole in the context of SDN are discussed in [3]. While integrated tests are a long term goal, it is necessary to understand the behavior of the individual network elements, i.e., controllers and switches in case of SDN. In an effort to provide means to test switch behavior with respect to compliance with the OpenFlow protocol specification, the authors of [4] present the OFTest suite. In contrast to this work, they focus on function tests rather than performance tests.

The study conducted in [5] features a dedicated hardware traffic generator in order to test the data plane performance of Linux-based OpenFlow switching. In a similar setup, the authors of [6] investigate the characteristics of virtual switches and underlying virtualization techniques. In both works, the main interest lies in the data plane performance of the different switch implementations. This work, on the other hand, investigates different measurement mechanisms for the control plane performance of OpenFlow-enabled switches and provides a first step towards classifying these mechanisms according to criteria like accuracy and complexity.

OFLOPS [7] is a software framework for testing OpenFlow switch performance in the data plane as well as in the control plane. Its extension, OFLOPS-Turbo [8] is capable of 10 GbE traffic generation and utilizes the open-source NetFPGA-based OSNT [9] traffic generator and capture system. Focusing on the processing of FlowMod messages, our work aims at comparing different mechanisms for switch performance evaluation and identifying trade-offs between them.

Furthermore, analytical approaches like [10] and [11] investigate the influence of various network parameters on the performance of an OpenFlow architecture. Since such models are often based on measurements, the accuracy of these measurements also positively affects the quality of the resulting models. Therefore, one key aspect of our analyses is the accuracy of the available measurement mechanisms. A methodology for assessing the accuracy of measurements in the SDN context is presented in [12]. In addition to measurements performed by an SDN controller module, wiretaps installed at both ends of a communication channel serve as a means of providing the ground truth. This technique is also applied in our experiments.

III. Methodology

In this section, two communication schemes for exchanging FlowMod and Barrier messages between the controller and the switch are described. Additionally, configuration and specification details of the measurement tools and OpenFlow switches used in this work are presented. Finally, the experimental setup is introduced alongside the measured parameters.

A. Methods of Sending FlowMod Messages

Information exchange between systems can be performed by means of one of two paradigms: *synchronous* and *asynchronous* messaging. Generally, asynchronous messages are passed between two entities: the sender emits multiple messages and does not wait for a response to continue sending the next messages. It contrasts with synchronous messaging, where the sender does not send a new message until it receives the response to the previous one. In the scenario of sending FlowMod messages, the OpenFlow specification [13] defines the optional Barrier messages which can be used in order to perform both kinds of messaging. The desired behavior can be achieved by using Barrier messages either after each FlowMod message or after multiple FlowMod messages. When installing rules on an OpenFlow switch, the OpenDaylight controller[1] offers implementations for those methods as described in detail in Fig. 1.

The time sequence diagram when applying the synchronous method to send FlowMod messages is presented on the right hand side of Fig. 1. The controller always sends messages in a sequence where a FlowMod message is followed by a Barrier Request message and waits for a Barrier Reply, before dispatching the next FlowMod. By doing this, the controller ensures that each requested rule is actually installed in the table of the switch. In this work, that method is referred to as *addFlow* and is distinguished from *addFlowAsync* - where the controller sets the rules by sending a batch of FlowMods and terminates the process with a Barrier Request. Whenever the switch finishes installing all the rules, it informs the controller by means of a Barrier Reply. Note, however, that using Barrier messages is not mandatory according to the OpenFlow protocol.

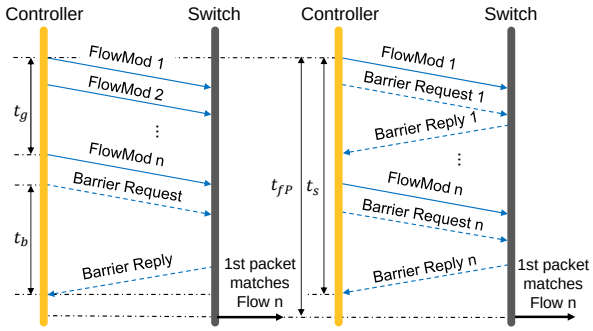


Fig. 1: Asynchronous and synchronous methods for adding flows to an OpenFlow switch.

Furthermore, Figure 1 illustrates the different components of the FlowMod processing time that are investigated in this paper:

- Time until the first packet t_{fP} - the time between the first FlowMod which is sent from the controller and the first data plane packet which matches the switch rule installed by the last FlowMod message.
- Setup time t_s - the latency between the first FlowMod and the last Barrier Reply.
- Generation time t_g - the time for generating n FlowMod messages (in case of addFlowAsync).
- Barrier time t_b - the delay between Barrier Request and Barrier Reply.

B. Measurement Tools and Devices Under Test

There are several options for assessing the FlowMod performance of an OpenFlow switch in terms of the aforementioned components of the processing time. Three such approaches are discussed in this work.

First, a purely software-based solution using the OpenDaylight controller is analyzed. In this context, the Hydrogen release[2] of the OpenDaylight controller is installed on a PC[3] which uses the Ubuntu 12.04 operating system. While the controller is attached to the OpenFlow switch under test and generates control plane traffic, i.e., FlowMod and Barrier messages, the Iperf[4] software running on an additional server is used to send UDP traffic to the switch's data plane. Finally, the traffic sink waits for the arrival of matched packets. In

[1] http://www.opendaylight.org/

[2] https://www.opendaylight.org/software/downloads/hydrogen-base-10

[3] Intel(R) Core(TM)2 Duo CPU E8500/4G RAM

[4] https://iperf.fr/

the context of the OpenDaylight controller, there are two options for obtaining the desired components of the FlowMod processing time. On the one hand, traffic captures recorded by the tcpdump tool running on the machine that hosts the controller can be used to calculate t_g, t_b, and t_s. On the other hand, it is possible to implement a Java module for the controller which intercepts the timestamps of events like sending FlowMods or receiving Barrier messages in order to derive the aforementioned times. Due to the use of concurrency within the implementation of these methods, however, not all timestamp-based measurements are as accurate as the ones obtained via tcpdump.

The second approach utilizes a Spirent C1[5], an Ethernet testing platform which allows generating traffic according to different protocols including OpenFlow. This device has four 1GbE interfaces, three of which emulate the controller, traffic source, and traffic sink, respectively. FlowMod messages are sent to the management port of the switch. Simultaneously, the traffic source keeps sending traffic to the traffic sink. The Spirent Test Center software (STC) provides means to measure traffic characteristics like packet delay, including values corresponding to t_g, t_b, t_s, and t_{fP}. In addition to the result database generated by the STC, the Spirent C1 also allows directly recording packet traces at each individual port. Using these captures, it is possible to determine the components of the FlowMod processing time as well.

Furthermore, the OFLOPS framework introduced in Section II provides means for measuring FlowMod processing times as well. However, at the time of writing, there is no publicly available version of OFLOPS which implements these measurement features. Therefore, we only assess the capabilities of the framework in terms of measuring FlowMod processing time components according to [7].

Table I shows the capabilities of the three tools in terms of measuring the different types of processing times that are discussed in this work. The tools include the Spirent Test Center software, an OpenDaylight controller module (ODLM), and the OFLOPS framework. Additionally, two wiretap devices are used to accurately capture the processing times which are used as ground truth.

TABLE I: Comparison of measurement mechanisms.

Time	**addFlowAsync**			**addFlow**		
	STC	ODLM	OFLOPS	STC	ODLM	OFLOPS
t_g	✓	✓	✓	N.A.		
t_b	✓	✓	✓			
t_s	✓	✓	✓	✓	✓	✓
t_{fP}	✓	✗	✓	✓	✗	✓

C. Measurement Setup

In order to evaluate the performance of an OpenFlow switch with respect to the processing time of FlowMod messages, the following scenario is implemented in a testbed. The testbed is set up according to recommendations for testing OpenFlow performance published by Spirent[6] and is shown in Fig. 2.

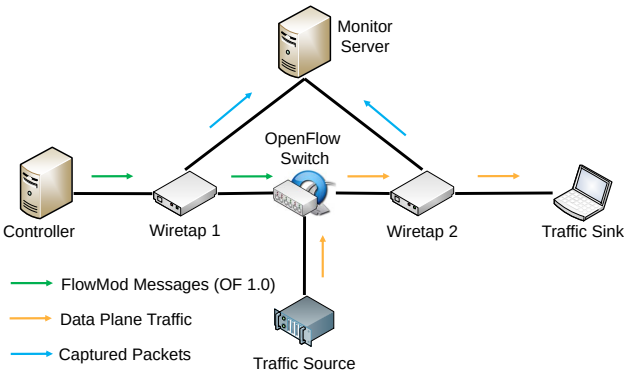


Fig. 2: Logical testbed setup.

Each experiment starts with an empty OpenFlow table in the switch. Then, the controller sends a FlodMod message that installs a rule with the lowest priority to drop all packets in order to avoid forwarding unmatched packets to the controller. Afterwards, the controller starts sending a number FlowMods to install these rules according to one of the two methods that were presented in Section III-A. Simultaneously, the traffic source generates data plane traffic that matches the last rule. Finally, the results of the experiment are extracted from the reports that are generated by the OpenDaylight controller or Spirent Test Center, allowing to evaluate how fast an OpenFlow switch processes FlowMod messages. The results are validated by using tap devices to capture transmitted packets and accurately calculate the latency between the first FlowMod and the arrival of the first packet which matched the last rule.

Two Net Optics tap devices[7] are installed before and after the OpenFlow switch in order to mirror both control plane traffic to and from the switch as well as data plane traffic to the traffic sink. The monitor server HP Proliant DL32 uses two Endace DAG 7.5G2 capture cards to capture every incoming packet. Based on their time stamps, it is possible to calculate all components of the FlowMod processing time as well as t_{fP} for validating the installation of OpenFlow rules. On the data plane, the traffic source sends UDP traffic with its IP as source IP address and the destination IP address corresponding to the traffic sink. This traffic can not reach the traffic sink until the last FlowMod, which has the matching fields regarding source and destination IP addresses, is installed.

In order to avoid possibly misleading results, the experiments are implemented using three OpenFlow switches from different vendors, focusing on an evaluation of the installation

[5]http://www.spirent.com/Test-solutions_datasheets/Broadband/PAB/Spirent_TestCenter/STC_C1-Appliance_Datasheet

[6]http://www.spirent.com/~/media/White%20Papers/Broadband/PAB/OpenFlow_Performance_Testing_WhitePaper.pdf

[7]http://www.ixiacom.com/products/ixia-gig-zero-delay-tap/

speed of OpenFlow rules. The specifications of the switches under test are summarized in Table II.

TABLE II: Switches used in this work.

Switch	CPU	Memory	Software
Pronto 3290	MPC8541 825 MHz	512MB	PicOs 2.0.14 (Open vSwitch v1.10.0)
Quanta T1048	MPC8541 825 MHz	1024MB	PicOs 2.6 (Open vSwitch v2.3.0)
NEC PF5240	PowerPC 667 MHz	1024 MB	OS-F3PA v5.0.0.1

The flow table of the NEC PF5240 has a limit of 2816 entries and the Quanta switch often exhibits unexpected behavior in the context of generating more than 2000 flow table entries. Hence, the number of FlowMod messages used in the experiments ranges from 2 to 1800 messages. Each run is repeated 10 times in order to obtain 90% confidence intervals.

IV. Results

This section provides the results of the experiments that were described in Section III. First, a comparison of the behavior of the different switches is presented by analyzing various components of the FlowMod processing time that are recorded at the wiretaps. Second, an in-depth evaluation of the accuracy levels achieved by the different measurement methodologies demonstrates their feasibility in the tested scenarios. Finally, recommendations regarding the choice of methodology are derived from aggregated measurement results.

A. Comparison of Switch Behavior

Figure 3 shows different components of the FlowMod processing time based on measurements performed at the wiretaps while using different numbers of FlowMod messages and different switches. In particular, the time between the first FlowMod and the last Barrier Reply, t_s, and the time until receiving the first packet on the data plane, t_{fP}, are presented. On the x-axis, the number of flows is displayed. According to the limitations of the different switches under test, this number is varied between 10 and 1800. Differently colored and shaped curves denote different switch models and time components, respectively. The y-axis represents the mean processing time that results from the corresponding parameter combination. Additionally, error bars provide 90 % confidence intervals obtained by repeating each experiment 10 times. For the presented measurements, FlowMod messages were generated with the OpenDaylight controller using the *addFlowAsync* method.

While the NEC switch exhibits the longest processing times of up to almost 5 seconds in case of installing 1800 flows, it is the only device for which the relationship $t_s > t_{fP}$ holds throughout all experiments. Thus, when the controller receives the Barrier Reply message, the switch's flow table update is already finished, resulting in a consistent state between switch and controller. For the Pronto and Quanta switches, on the other hand, this relationship is reversed: after receiving the Barrier Reply message, the controller expects the flow installation to be completed, although it is still taking place in the switches. Such an inconsistency could potentially cause unexpected behavior, e.g., when the controller uses its northbound API to communicate the seemingly finished update to an application that starts its transmission immediately. While the extent of this deviation is in the order of magnitude of 50 ms in case of Pronto, it equals roughly 400 ms in case of Quanta. On the other hand, the Quanta switch consistently outperforms the other two models in terms of t_s and t_{fP} by a significant margin as soon as more than 600 flows are installed.

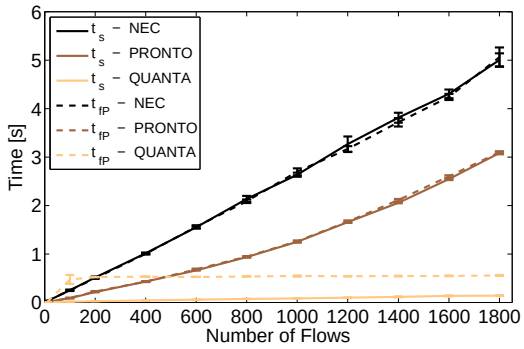


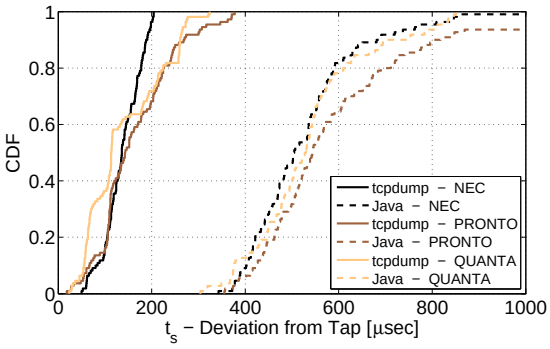
Fig. 3: Different switch behavior measured at the wiretaps.

These results demonstrate that each switch model might have its own characteristic behavior. Hence, network operators need to evaluate the performance of hardware devices before deploying them in the network in order to ensure a reliable network behavior. In the following sections, the accuracy of different approaches for measuring switch characteristics is evaluated.

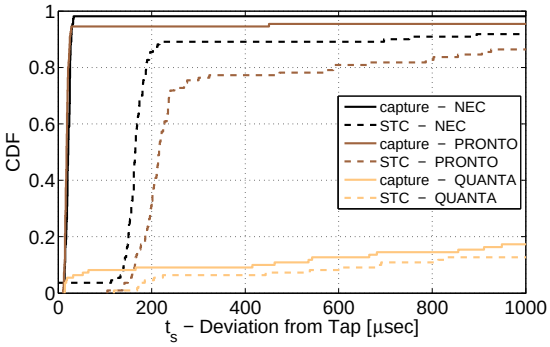
B. Accuracy Assessment of Measurement Mechanisms

The setup time t_s is an important performance indicator for OpenFlow switches since it usually constitutes the majority of the FlowMod processing time. Additionally, in case of the OpenDaylight Java module, it is possible to measure the value of this parameter during runtime and integrate it into the controller's feedback loop. Therefore, Figure 4 displays cumulative distributions of the accuracy of the t_s measurements for the two measurement tools OpenDaylight and Spirent C1. In this context, the accuracy refers to the difference between the value recorded by the measurement tool and the corresponding wiretap which is considered to be the ground truth. There is also a distinction between results from different measurement probes, i.e., Java and tcpdump in case of OpenDaylight and Spirent Test Center and port-based captures in case of the Spirent C1.

Figure 4a shows the accuracy of the measurement probes available for the OpenDaylight controller. While the accuracy in microseconds is displayed on the x-axis, the y-axis denotes the fraction of measurements that are below a particular



(a) OpenDaylight module.



(b) Spirent C1.

Fig. 4: Distribution of the accuracy of the t_s measurement when using different measurement tools and probes.

accuracy threshold. Different line colors represent different switches and line shapes correspond to measurement probes. In case of OpenDaylight, curves corresponding to different probes form two groups whose values differ only marginally from switch to switch. The first group represents the tcpdump measurements and features values that are mainly in the range between 100 and 300 μsec. The values reported by the Java module inside the controller are higher, with values ranging from 400 μsec to 1 ms. This behavior is consistent with the measurement setup: since the Barrier Reply message from the switch first passes the controller's network interface before arriving in the user space Java application, the time measured by the latter is higher than in case of tcpdump.

Like the previous figure, the curves in Fig. 4b form groups according to the measurement probe. While the port-based capture provides measurements that are accurate up to an order of magnitude of roughly 30 μsec most of the time, the reports generated via the Spirent Test Center show a difference of around 200 μsec as well as significantly more outliers in terms of accuracy. When using the Spirent C1 in conjunction with the Quanta switch, irregular behavior is observed regardless of the number of installed flows. Since the Spirent C1 is a proprietary closed-source system, this phenomenon can not be investigated in detail.

While Figure 3 shows that t_s and t_{fP} are almost identical for the majority of switch models, Figure 5 presents a quantitative view on the observed deviation between tool-based measurements of the setup time t_s^{tool} and the time until the first packet t_{fP}^{tap} as reported by the wiretap. Values on the x-axis represent the difference $\delta = t_s^{tool} - t_{fP}^{tap}$ and the y-axis indicates the fraction of measured values below each threshold. Due to the irregularities discussed in the previous paragraphs, measurements regarding the Quanta switch are omitted for the sake of readability. As reported before, the tcpdump and Java probes provide very similar data, with tcpdump being slightly more accurate. Additionally, the figure allows deriving guidelines for determining the time until a set of FlowMod messages are actually processed in the data plane of a particular switch model. For example, $\delta > 0$ in case of the NEC switch implies that the Barrier Reply is always sent after the rule is installed in the switch. Hence, the controller receives information that corresponds to the switch's data plane. In contrast, the minimum value of $\delta = -170$ ms in case of the Pronto switch means that up to 170 ms may pass before the controller and switch are in a synchronized state. Information regarding these delays can be obtained by performing such measurements before actually deploying the switches in a network. Hence, operators can incorporate this information into the northbound and southbound APIs of the SDN controller and increase the reliability of the information exchange in the network.

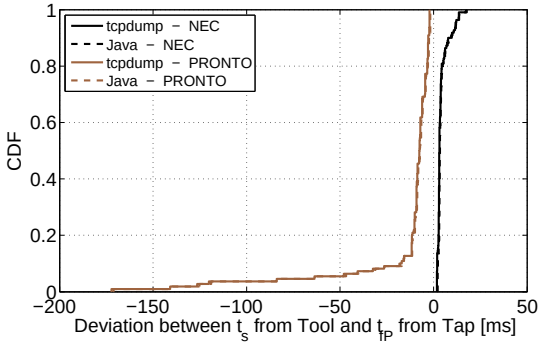


Fig. 5: Distribution of $\delta = t_s^{tool} - t_{fP}^{tap}$.

C. Correlation Analysis of Measurement Mechanisms

In order to provide an aggregated overview of the results, Table III shows the correlations between the tool-based measurements and the ground truth according to the wiretap devices. Previous results indicate that in case of the Spirent C1, the port-based captures provide a higher level of accuracy. In the context of the OpenDaylight controller, the Java module is more relevant in a practical context due to its capability to perform measurements during runtime. Hence, only these probes are included in the table. The correlation is determined according to Spearman's rank correlation coefficient.

In addition to the correlation between measurements of the same component of the FlowMod processing time, the

TABLE III: Correlations between measurements from different tools and the wiretap-based ground truth.

Mode	addFlowAsync					addFlow	
Tool	ODL/Java		Spirent/capture			ODL/Java	Spirent/capture
Switch \ Pair	(t_s^{tap}, t_s^{tool})	$(t_{fP}^{tap}, t_s^{tool})$	(t_b^{tap}, t_b^{tool})	(t_g^{tap}, t_g^{tool})	(t_s^{tap}, t_s^{tool})	$(t_{fP}^{tap}, t_s^{tool})$	$(t_{fP}^{tap}, t_s^{tool})$
NEC	1.0000	0.9999	0.9983	0.9999	1.0000	0.9999	0.9988
Pronto	0.9999	0.9980	0.9958	1.0000	0.9992	0.9994	0.9989
Quanta	0.9999	0.8146	0.8790	0.9997	0.9984	0.9396	0.3727

table also provides information on the relationship between t_s measured via the tools and t_{fP} measured via the wiretap. A high degree of correlation in this context implies that it is possible to reliably predict the time until the requested FlowMod messages are installed in the switch's data plane when the t_s measurements are given.

For the NEC and Pronto switches, all correlations are above 99 %. This behavior is in line with previous observations which show deviations in the order of magnitude of less than 1 ms for values that are as high as multiple seconds. In case of the Quanta switch, however, there is no significant correlation between t_s and t_{fP}. As observed in Fig. 3, the time until the first data plane packet is matched in the switch is nearly constant while the setup time increases when the number of installed flows is increased.

V. Conclusion

This work presents and compares two mechanisms for evaluating the performance of OpenFlow switches in terms of processing FlowMod messages. On the one hand, we use a software-based approach featuring a module for the OpenDaylight controller implemented in Java. On the other hand, the Spirent C1 dedicated testing platform is utilized. The two mechanisms are evaluated with respect to the accuracy of their measurements of several components of the FlowMod processing time. Furthermore, they represent different classes of mechanisms that are available to network operators who need to make sure that a planned SDN deployment meets the requirements of a particular use case. The aspect of switch performance evaluation is especially relevant since our measurements show significant differences between different switch models.

Results of the experiments with three switches from three different vendors show that both mechanisms achieve an accuracy in the sub-millisecond range when compared to measurements performed with dedicated capture cards at wiretaps. Except for one switch model with unexpected behavior, the mechanisms achieve similar accuracy levels independent of the device under test. Furthermore, high correlations between measurements at the tools and the wiretaps indicate that measured values can be used to derive performance measures even more accurately.

In addition to the benefits regarding costs and ease of use, the Java-based controller module also has the capability to run during normal operation. This enables features like switch performance monitoring at low costs in terms of resource overhead. Finally, we observe a high correlation between the time until receiving the Barrier Reply message and the time until the rule installation in the switch is actually completed. Therefore, it is possible to increase the state consistency between controller and switch by utilizing such a controller module.

Acknowledgment

This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS and is partly funded by the BMBF. The authors alone are responsible for the content of the paper.

References

[1] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, Attributes, and Use Cases: A Compass for SDN," *Communications Magazine, IEEE*, 2014.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM CCR*, 2008.

[3] M. Kuźniar, M. Canini, and D. Kostić, "OFTEN testing OpenFlow networks," in *European Workshop on Software Defined Networking (EWSDN)*, 2012.

[4] "OFTest – Validating OpenFlow Switches," Big Switch Networks. [Online]. Available: http://www.projectfloodlight.org/oftest/

[5] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "Openflow switching: Data plane performance," in *IEEE International Conference on Communications (ICC)*, 2010.

[6] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance characteristics of virtual switching," in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014.

[7] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An open framework for OpenFlow switch evaluation," in *Passive and Active Measurement*, 2012.

[8] C. Rotsos, G. Antichi, M. Bruyere, P. Owezarski, and A. Moore, "OFLOPS-Turbo: Testing the Next-Generation OpenFlow switch," in *European Workshop on Software Defined Networks (EWSDN)*, 2014.

[9] M. Shahbaz, G. Antichi, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. Feamster, N. McKeown, B. Felderman, M. Blott *et al.*, "Architecture for an open source network tester," in *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*, 2013.

[10] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *Proceedings of the 23rd international teletraffic congress*, 2011.

[11] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou, "An analytical model for software defined networking: A network calculus-based approach," in *IEEE Global Communications Conference (GLOBECOM)*, 2013.

[12] M. Jarschel, T. Zinner, T. Höhn, and P. Tran-Gia, "On the Accuracy of Leveraging SDN for Passive Network Measurements," in *Australasian Telecommunication Networks & Applications Conference (ATNAC)*, 2013.

[13] "OpenFlow Switch Specification v1.5.0," Open Networking Foundation, 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf