On the Usability of OpenFlow in Data Center Environments

Rastin Pries, Michael Jarschel, Sebastian Goll
University of Würzburg, Institute of Computer Science, Würzburg, Germany.
Email: {pries,michael.jarschel,goll}@informatik.uni-wuerzburg.de

***Abstract*—Since its introduction OpenFlow has been used as an enabler for network experiments in a variety of fields. Although OpenFlow was initially only used in the research domain, the concept is now finding its way into data centers due to the relatively cheap hardware and high flexibility. In this paper, we take a look at the scalability and usability of OpenFlow in data centers. Based on data center traffic models and OpenFlow measurements, we evaluate whether OpenFlow is able to cope with the short flow inter-arrival times of the traffic models by means of simulation.**

***Index Terms*—OpenFlow, data center, performance evaluation**

I. Introduction

Data centers are attracting more and more interest, offering a large variety of services such as online gaming, data storage, data processing, and online office products. However, there are still a lot of challenges to be solved, e.g., overall performance, energy efficiency, resilience, scalability, and how to transport the data to the consumer. Most data center providers currently focus on building their data centers only with commercial off-the-shelf (COTS) hardware to reduce the cost and to be easily maintainable. In addition, the data center should be extensible and should scale up to 100,000 servers.

Often neglected is the data center network architecture, which has a large influence on both, the energy consumption as well as the service times [1]. In addition to the COTS servers, we can also see a shift from costly layer-3 switches with hundreds of ports to COTS layer-2 switches. These switches have however to be able to handle the huge amount of traffic and should still be flexible enough. This is where OpenFlow comes into play. OpenFlow was first proposed by McKeown et al. in 2008 to enable researchers to conduct experiments in production networks [2]. Meanwhile, OpenFlow is being used beyond research for enabling network virtualization in data centers, e.g. at CERN. OpenFlow offers a higher flexibility compared to classical switches as it performs a flow-based switching instead of packet-based and it separates the control plane from the data plane. This allows to set up a separate controlling entity, the OpenFlow controller, responsible for changing forwarding rules of a number of switches.

OpenFlow is currently standardized by the Open Networking Foundation (ONF) [3], whose members are in addition to classical operators also owners of large data centers such as Google and facebook. As a result, we can expect a growing number of works conducting experiments in OpenFlow-enabled data centers. In these data centers, OpenFlow is used to evaluate new virtualization techniques or new routing protocols. However, few performance evaluations of the OpenFlow architecture exist, e.g. if OpenFlow scales well and if it can be used in large data centers.

In Jarschel et al. [4], we set up a basic model for evaluating the performance of OpenFlow in a simple scenario. The model is based on results from queuing theory and is verified by simulations and measurement experiments with an OpenFlow switch and controller. In this paper, we extend the simulation from [4] to be able to simulate the edge switches of a data center. The traffic is thereby generated based on data center measurements published by Benson et al. [5]. Due to the complex structure of the simulation and the data center traffic models, we are unfortunately not able to extend our analytical model to verify the simulation results.

The remainder of the paper is structured as follows. In Section II, we introduce the OpenFlow architecture in more detail and provide an overview on related work. Section III describe our simulation scenario and shows some measurement results needed a input parameters for the simulation. The results from the performance evaluation are presented in Section IV. Conclusions are drawn in Section V and a brief outlook on future work is given.

II. Background and Related Work

In this section, we first give a brief introduction to OpenFlow, before presenting the work related to OpenFlow performance. More details on OpenFlow can be found in the white paper [2] as well as in the OpenFlow specification [6].

OpenFlow was designed in such a way that it enables researchers to test new ideas under realistic conditions on an existing network infrastructure. To be able to take action in the switching, OpenFlow separates the control plane from the data plane and connects them by an open interface, the OpenFlow protocol. The control plane is implemented in software in form of a controller on an external computer. The communication between the switch and the controller is realized over a secure channel. The separation of the control plane enables researchers to flexibly test new algorithms on high-performance hardware without influencing the production traffic. In the following, we describe the flow handling as

This work was partially funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01BK0917, GLab). The authors alone are responsible for the content of the paper.

described in the OpenFlow 1.0 specification. Although the OpenFlow specification 1.2 has already been released, we concentrate on version 1.0 as it is the newest version supported by most OpenFlow implementations.

Each OpenFlow switch holds a flow entries which stores three components, header fields, counters, and actions. Using OpenFlow 1.0, 12 header fields are extracted from a packet header and matched with the flow entries. If the packet does not match on of the flow entries, it is forwarded to the controller, which determines how the packet should be handled. If the packet matches a flow entry, the counters are updated and the appropriate actions are performed. Counters are used for statistical reasons to store e.g. the number of received packets per flow, the duration of the flow, etc. The action field in the flow table describes how the switch handles incoming packets. The packet handling process is visualized in Figure 1.

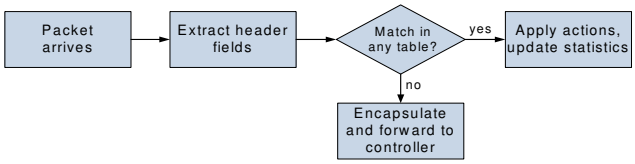

Fig. 1. Packet handling in an OpenFlow switch.

Most papers published on OpenFlow [7]–[10] focus on demonstrating the concept of splitting the control plane from the data plane in various fields. Heller et al. [7] show how OpenFlow can be used to save energy in data centers, while others focus on route optimization or network virtualization. However, non of these papers focuses on performance issues of OpenFlow.

There are two OpenFlow benchmarking tools available, called OFlops [11] and Cbench [12]. OFlops can gauge the datapath performance by measuring switch response times in a variety of scenarios. Cbench can either measure the controller throughput or processing delay in individual tests. Both tools are as of yet considered experimental.

Bianco et al. [13] measure the OpenFlow switching performance for different types of rules and packet sizes. A similar approach is taken by Sünnen in his student thesis [14]. The author analyzes the performance impact of the flow table in various scenarios. Both works mainly focus on the datapath.

The work with the closest relation to our paper is presented by Curtis et al. [15]. According to them, the current OpenFlow implementations do not meet the requirements of today's data centers as several microflows in data centers create excessive load on controller and switches. The proposed DevoFlow reduces the communication between switches and controller as the controller is not invoked for every flow setup.

In contrast to the above mentioned papers, we focus on evaluating the performance of OpenFlow in a data center environment using data center traffic models gathered from real data center measurements by Benson et al. [5].

III. OpenFlow Measurements & Simulation Scenario

Before describing our simulation and the considered scenario, we first derive the performance parameters. Therefore, we set up an OpenFlow testbed and measure the forwarding performance of various OpenFlow switch implementations.

A. Measurements

To get the input parameters for the simulation, we set up an OpenFlow testbed as depicted in Figure 2. A server is used as load generator to test the performance of the attached OpenFlow switch implementations. The OpenFlow switch itself is connected to the OpenFlow controller as well as to two wire taps. The taps enable us to measure the packets before and after they are processed by the OpenFlow switch. The measurement server itself is equipped with an Endace DAG card to keep the measurement error below nine nanoseconds [16]. For our experiments, we use three different OpenFlow implementations, a Pronto 3290 24-port gigabit switch, a NetFPGA OpenFlow switch, and the Open vSwitch software switch.

To study the switch performance without any controller interaction, we pre-install a flow rule in the switch and send packet bursts of two million identical packets through the switch, which match the rule. The packet size is varied between 64 Byte and 1514 Byte Ethernet frame length to see the impact on the switch performance. Figure 3 presents the measured forwarding delays in microseconds for all three switches depending on the packet payload. The results are averaged from 10 independent measurement runs and the error bars indicate the 95% confidence level. The y-axis is scaled logarithmically to be able to display the curves for all switches in one figure.

Both hardware implementations show an almost linear increase of the mean forwarding delay from about $4\mu s$ to $16.5\mu s$ with the NetFPGA performing slightly better. The performance of the software implementation, the Open vSwitch is two orders of magnitude slower because the software switch suffers from frequent memory accesses, especially at small packet

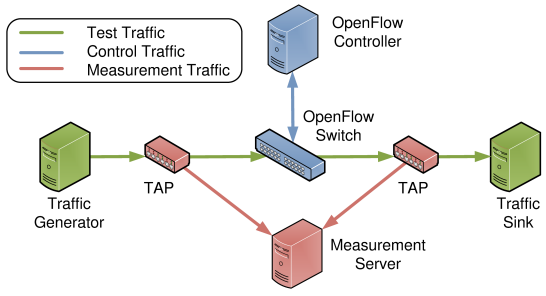

Fig. 2. Testbed set up to measure model parameters.

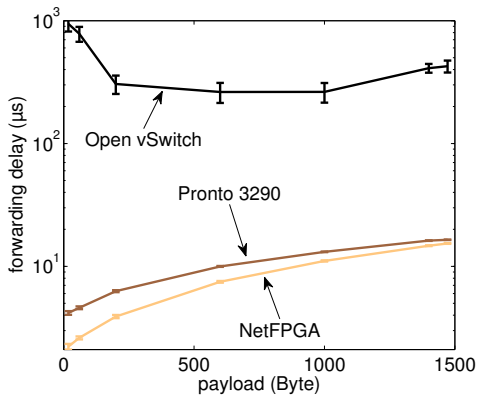

Fig. 3. Measured switch delay.

TABLE I
SIMULATION PARAMETERS.

Entity	Parameter	Value
Controller	queue size	1024 slots
	mean processing time	$240\mu s$
	distribution of processing time	negative exponential
Switch	number of switches	8
	mean processing time	$9.82\mu s$
	distribution of processing time	Erlang-25
	distribution of flow inter-arrival time	$PRV2_1, PRV2_2$, $EDU1$ from [5]
	relative flow inter-arrival time	inter-arrival time distr. scaled between 0 and 2
Simulation	duration	10 million successfully transmitted flows
	repetitions	6 for each parameter

sizes. For our simulation, we use the switch delay of the Pronto switch because its hardware is closest to the switches, which are used at the edge layer of a data center. Thus, we use a negative exponential distributed switching delay with a mean of $9.82\mu s$ for the simulation runs.

To measure the performance of an OpenFlow controller, we use Nox 0.9 as controller and installed the Cbench [12] tool on the measurement server and attach the measurement server directly to the controller. Cbench measures the rate in which flow requests are handled by the controller. Our measurements show an average rate of 4175 responses per second with a standard deviation of 101.43. From this value, we calculated the mean values of the controllers sojourn times and use these for the simulation, cf. Table I.

B. Simulation Setup

To simulate the performance of OpenFlow in a data center environment, we use the OMNeT++ simulation environment and set up the most common data center topology, which is depicted in Figure 4. It consists of three different layers, the access layer, the aggregation layer, and the core layer. The routers of the core layer are attached to the transport network on one side and to the aggregation layer on the other side. The aggregation layer facilitates the increase in the number of server nodes (more than 10,000 servers) while keeping inexpensive layer-2 switches in the access network for providing a loop-free topology. In this paper, we focus on the layer-2 switches at the edge layer.

We assume a total of eight edge OpenFlow switches similar to the figure. The switches are connected to a single OpenFlow controller via a separate link. As soon as a packet of a new flow is received at one of the switches, the packet is transmitted to the controller and the controller installs the flow rule on every switch the flow traverses. The flow rule is deleted at the switch as soon as the idle timeout expires. Thus, we follow a "reactive" approach, meaning that no flow or wildcard rules are pre-installed.

For a realistic traffic model at the OpenFlow edge switches, we use measurements published by Benson et al. [5]. They measured flow inter-arrival times on data center edge switches, both within private and university data centers. We use a selection of these measurements to simulate the performance of OpenFlow. Sticking to the abbreviations introduced by Benson et al., the distributions that we investigate are EDU1, $PRV2_1$, and $PRV2_2$. Figure 5 shows a reproduction of the distributions in question, represented by their cumulative distribution functions of flow inter-arrival times. The best and the worst case scenarios are EDU1 and $PRV2_1$ respectively. The university data center (EDU1) seems to be rather low loaded compared to the private data center (PRV2). In addition to the highest loaded edge switch in PRV2, we also use the values gathered from another edge switch in the same data center, namely $PRV2_2$, which represents an average loaded switch measured by Benson et al. [5].

All parameter used for the simulation runs are listed in Table I. We also evaluated the performance using different distributions for the switch as well as the controller processing time, but as the general behavior is similar, we omitted these results.

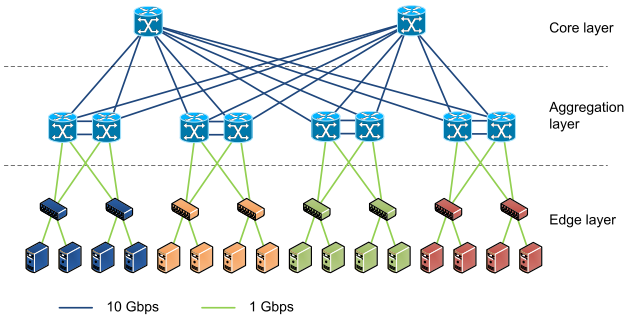

Fig. 4. Three-tier data center topology.

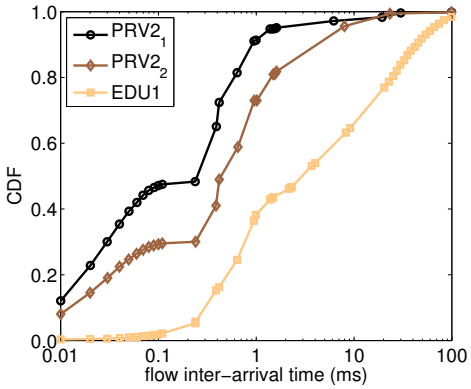

Fig. 5. Flow inter-arrival times from Benson et al. [5].

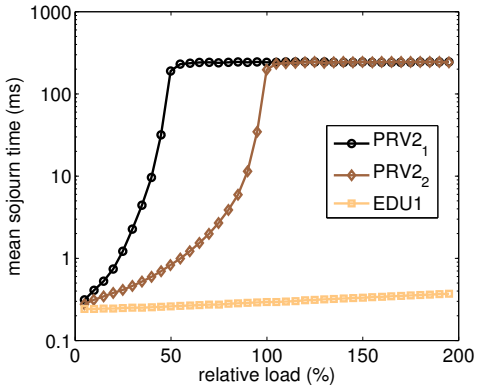

Fig. 6. Sojourn time for the three-tier data center.

IV. Performance Evaluation

First, we want to evaluate the sojourn times and its Coefficient of Variation (CV) using the three different flow inter-arrival times from Figure 5. We thereby linearly scale the distributions, to put more or less load on the OpenFlow controller. Figure 6 shows the sojourn time and the variation in sojourn time is plotted in Figure 7. 100% on the x-axis represents the original, unmodified distribution and does not necessarily imply a 100% controller load. It has to be mentioned that the y-axis in Figure 6 is plotted using a logarithmic scale. As we measured only a small difference between the six different simulation runs, we did not plot the confidence intervals.

As can be seen, the three distributions show significantly different results with regard to the OpenFlow controller performance. In the case of EDU1, the sojourn time is approximately the same over the entire range of loads, i.e. ranging from zero to twice the original flow rate. Similarly, the CV of the sojourn time, see Figure 7, is nearly constant at 1.0. The reason for this is the relatively large flow inter-arrival time given by the original measurements with a median value of approximately 3 ms. In the case of the EDU1 distribution, the single OpenFlow controller is sufficient to support the eight edge switches in the network.

This is further supported by Figures 8 and 9 which illustrate the controller queue length as well as the observed packet loss. A maximum controller queue length of about 30 and a mean controller queue length well below 0.5 for all loads, as well as no packet loss at all, underline that OpenFlow can be used in data centers following a similar traffic distribution as EDU1.

On the other hand, the two private networks $PRV2_1$ and $PRV2_2$ put significantly more load on the OpenFlow controller. The original measurements done by Benson et al. give a median flow inter-arrival time of 0.25 ms for the $PRV2_1$ network and 0.44 ms for the $PRV2_2$ network, as indicated by the distribution shown in Figure 5. This significantly smaller flow inter-arrival time leads to proportionally more packets arriving at the OpenFlow controller, compared to the EDU1 network. The result of this is severe packet loss starting at 50% relative load for the $PRV2_1$ network, and 100% for the $PRV2_2$ network, as shown in Figure 9. This loss happens as soon as the controller queue fills up to the maximum size because more packets arrive at the controller than the controller can process per time unit. However, before this limit is reached, the mean queue sizes already start to increase, leading to severe delays and variations in the sojourn time through the controller, as some packets have to wait for a long time before they are processed.

The exact behavior, particularly of the CV of the sojourn time, highly depends on the specific distribution of flow inter-arrival times. In general, the CV first increases when the controller queue starts to fill up, and then suddenly decreases to nearly zero when severe packet loss is experienced. This reduction to a quasi-deterministic resulting traffic flow is due to the fact that every packet of a new flow that is not dropped has to wait until approximately 1024 other flow rules are installed before it is finally processed. The sojourn time distribution through the controller basically becomes the sum of 1024 Markov distribution, in other words, an Erlang-1024 distribution with a resulting small CV of approximately 0.03.

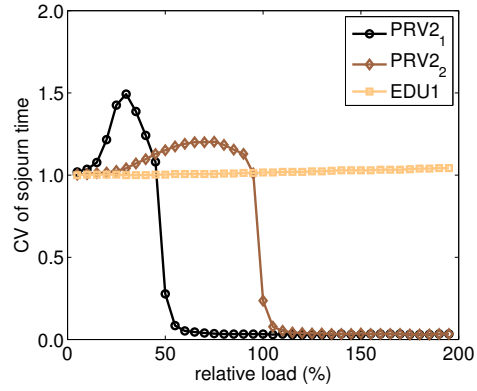

Fig. 7. Coefficient of variation of the sojourn time.

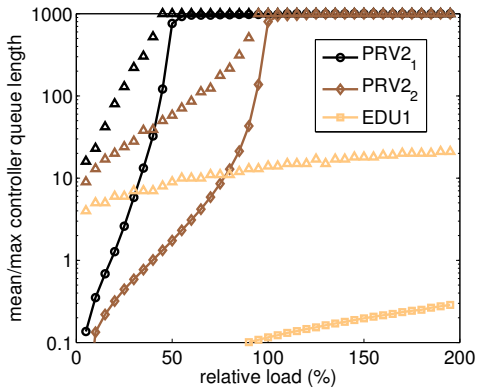

Fig. 8. Controller queue length. Triangles show the maximum queue length and the curves represent the average controller queue length.

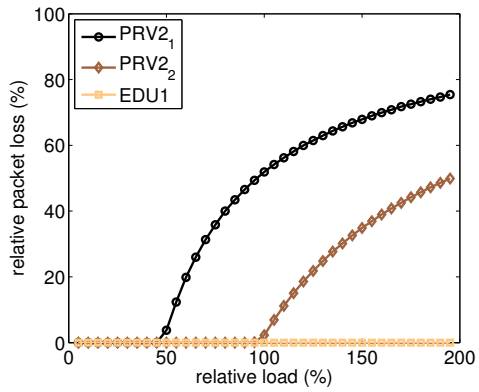

Fig. 9. Observed packet loss.

In summary, the simulated OpenFlow controller with an expected processing time of $240\mu s$ is more than sufficient for the EDU1 network on the one hand, but it is unable to cope with the requirements of the two private networks on the other hand. For these, a more powerful OpenFlow controller with an improved processing time would be necessary or the load could be split accross different controllers. It follows that a careful investigation is necessary before OpenFlow can be deployed in data center-like networks, so that the OpenFlow controller does not become the bottleneck of the entire system.

V. Conclusion

In this paper, we evaluated the usability of OpenFlow in a data center environment. In doing so, we verified that great care is required when planning OpenFlow systems, with regard to the resulting load on the OpenFlow controller in the system. We have seen that two out of the three data center measurements taken by Benson et al. [5] would require more than a single OpenFlow controller or severe data loss and high sojourn times for all non-lost packets would result for the Controller/Switch ratio of 1:8. Only the educational data center network would be suitable to immediate transformation into an OpenFlow setup with only a single OpenFlow controller. When still relying on a single OpenFlow controller, the flow matching rules have to be pre-installed and the controller should just be used for optimizing the data center performance, e.g. by optimizing the routing [17]. Possible directions for future research might be to evaluate different controller architectures so that even large data centers can be operated with OpenFlow.

Acknowledgments

The authors would gratefully thank Phuoc Tran-Gia from the University of Wuerzburg for the fruitful discussions and support on this paper.

References

[1] R. Pries, M. Jarschel, D. Schlosser, M. Klopf, and P. Tran-Gia, "Power Consumption Analysis of Data Center Architectures," in *GreenNets*, Colmar, France, October 2011.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008.

[3] ONF, "Open Networking Foundation," https://www.opennetworking.org/, 2011.

[4] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and Performance Evaluation of an OpenFlow Architecture," in *23rd International Teletraffic Congress (ITC 2011)*, San Francisco, CA, USA, September 2011.

[5] T. Benson, A. Akella, and D. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Internet Measurement Conference (IMC)*, Melbourne, Australia, November 2010.

[6] "OpenFlow Switch Specification, Version 1.2," https://www.opennetworking.org, December 2011.

[7] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastic Tree: Saving Energy in Data Center Networks," in *7th USENIX Symposium on Networked System Design and Implementation (NSDI)*, San Jose, CA, USA, April 2010, pp. 249–264.

[8] M. Casado, D. Erickson, I. A. Ganichev, R. Griffith, B. Heller, N. McKeown, D. Moon, T. Koponen, S. Shenker, and K. Zarifis, "Ripcord: A Modular Platform for Data Center Networking," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-93, June 2010.

[9] S. Das, G. Parulkar, P. Singh, D. Getachew, L. Ong, and N. McKeown, "Packet and Circuit Network Convergence with OpenFlow," in *Optical Fiber Conference (OFC/NFOEC'10)*, San Diego, CA, USA, March 2010.

[10] R. Braga, E. S. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," in *35th Annual IEEE Conference on Local Computer Networks*, Denver, CO, USA, October 2010, pp. 416–423.

[11] R. Sherwood and K.-K. Yap, "OFlops Switch Benchmark," http://www.openflowswitch.org/wk/index.php/Oflops, 2011.

[12] ——, "Cbench Controller Benchmarker," http://www.openflowswitch.org/wk/index.php/Oflops, 2010.

[13] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in *IEEE ICC*, Cape Town, South Africa, May 2010.

[14] D. Sünnen, "Performance Evalulation of OpenFlow Switches," Semester Thesis at the Department of Information Technology and Electrical Engineering, ETH Zürich, February 2011.

[15] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, August 2011. [Online]. Available: http://doi.acm.org/10.1145/2043164.2018466

[16] Endace, "DAG 7.5G2 Datasheet."

[17] M. R. Nascimento, C. E. Rothenberg, C. N. A. Correa, S. C. de Lucena, and M. F. Magalhaes, "Virtual Routers as a Service: The RouteFlow Approach Leveraging Software-Defined Networks," in *6th International Conference on Future Internet Technologies 2011 (CFI 11)*, Seoul, Korea, June 2011.