perfbench: A Tool for Predictability Analysis in Multi-Tenant Software-Defined Networks

Andreas Blenk[1] Arsany Basta[1] Laurenz Henkel[1] Johannes Zerwas[1]
Wolfgang Kellerer[1] Stefan Schmid[2]
[1] Technical University of Munich, Germany [2] University of Vienna, Austria

ABSTRACT

Network Virtualization (NV) provides low resource costs and high utilization, while ensuring bandwidth isolation in the data plane. Software-Defined Networks (SDNs) are a particularly interesting technology to implement NV, as tenants maintain control over their virtual Software-Defined Networks (vSDNs). However, bandwidth isolation alone may not be enough to provide a predictable application performance in virtual networks, as the virtualization layer itself is another source of potential performance interference. Today, we lack tools that help reveal and investigate such sources of interference and identify performance inefficiencies. In order to fill this gap, we developed a new tool — *perfbench*. We report on the tool design and our initial findings.

CCS CONCEPTS

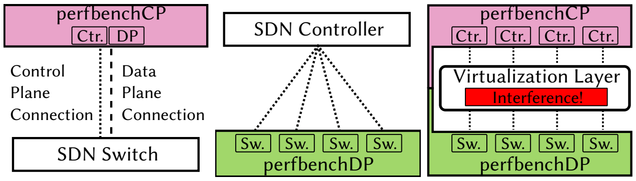
• **Networks → Network performance analysis**; *Network measurement*;

KEYWORDS

OpenFlow, Measurements, Performance, Multi-Tenancy, Network Slicing, Virtual Networks

1 INTRODUCTION

The Motivation: Controllable Virtual Networks. Network Virtualization (NV) enables a high degree of resource sharing with guaranteed bandwidth isolation per virtual network on the data plane. Software-Defined Networks (SDNs) provide a particularly interesting framework for network virtualization [2, 6]: a virtual SDN (*vSDN*) network offers great management flexibilities to its tenants. In fact, Google's Andromeda cloud infrastructure already deploys virtual networks using OpenFlow (OF) for different applications (which communicate only inside their virtual networks) [3].



(a) Switch. **(b) Controller.** **(c) Multi-tenant.**

Figure 1: OpenFlow benchmark tool *perfbench* and its three modes of operation. One tool for either benchmarking controllers, or for benchmarking switches, or when emulating both switches and controllers to detect interference in multi-tenant networks. Note Fig. (c) where tenants can interfere on the virtualization layer when competing for processing resources.

A major objective of any virtual network is to provide isolation, also in terms of performance: measurement studies have demonstrated that in order to provide a predictable cloud application performance, it is crucial to account for the potential interference *on the network* [4]. As applications such as batch processing, streaming, and scale-out databases, generate a significant amount of network traffic and as a considerable fraction of their run time is due to network activity, resource isolation needs to be ensured at any time.

The Problem: Detecting, Modeling and Mitigating Control Plane Interference. At the heart of any network virtualized architecture lies a *network virtualization layer (VL)* which multiplexes different tenants across a shared network infrastructure like data centers. In vSDNs, the layer is not only responsible for network abstraction, but also for control plane translation, i.e., managing the shared control of their respective virtual networks. Interestingly, however, not much is known today about the *performance impact* when sharing the network control. This is problematic, as the performance predictability of a given cloud application can only be as good as its least predictable component like the control plane part. A cloud application based on a model which ignores certain components (e.g., the VL and its impact on the control plane) entirely, may perform in unexpected ways.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCOMM Posters and Demos '18, August 20–25, 2018, Budapest, Hungary
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5915-3/18/08…$15.00
https://doi.org/10.1145/3234200.3234234

Contributions: This poster makes two contributions: (1) it presents *perfbench*, a benchmark tool for software-defined networks which supports high control traffic rates; (2) it makes the case for control plane performance studies by presenting measurement results shedding light on control plane interferences in multi-tenant SDN environments.

2 *PERFBENCH*: A SWISS-ARMY KNIFE FOR PERFORMANCE EVALUATION

While many SDN performance benchmark tools already exist, our design goal of *perfbench* was to make it a "swiss-army knife". Its advantage: being able to benchmark switches either as a controller (see *perfbenchCP* in Fig. 1a) or to benchmark controllers by emulating switches (see *perfbenchDP* in Fig. 1b), or to benchmark VLs where it emulates simultaneously multiple controllers and switches as demanded by multi-tenant environments (Fig. 1c). Here, one program consisting logically of the two parts *perfbenchDP* and *perfbenchCP* wraps the layer. Generally, the capability of *perfbench* to benchmark switches has already successfully been used in [1], where measurements revealed the latencies of OF messages and the CPU utilization of switches depending on constant OF message rates and types.

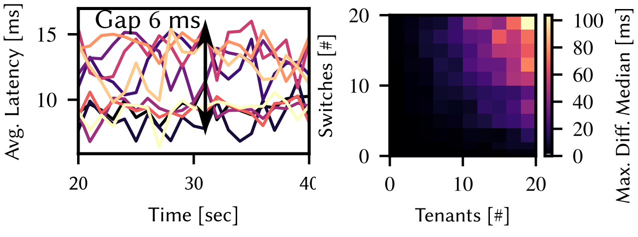
In order to identify potential control plane interference in multi-tenant setups, *perfbench* is tailored towards high throughput. It provides stable streams of different OF message types whose inter-arrival times and burstiness can follow various statistical distributions. It is built on top of the libfluid C++ library [7] and it is publicly available [5].

3 CASE STUDY: MULTI-TENANCY

In order to demonstrate *perfbench*'s capabilities and also to shed light on potential interference problems in multi-tenant SDN networks, we run different setups where *perfbench* emulates multiple switches and controllers while connecting to a VL (see again Fig. 1c). We would expect that a VL without any interference should lead to equal distributions of the control plane latencies perceived by each SDN controller.

In detail, we conduct the following case study: we measure 110 setups, i.e., the cross product of $1, 2, 4, \ldots, 20$ switches in combination with $2, 4, \ldots, 20$ controllers with FlowVisor serving as the VL; for instance, when 20 switches and 20 controllers are used, the VL manages 400 control plane connections. In each setup, each tenant sends the same amount of `OF_FLOW_MOD` messages per second, while the total amount of messages per second per setting is always 2 000; we measure the latency of every message. In our setup, we use two machines, one running *perfbench* and one hosting the VL.

As an example for one run, Fig. 2a depicts control plane latency averaged every second over time. Here, the latency gap among 10 tenants is quite large: the pronounced gap between the best and the worst tenant is 6 *ms*. When taking into account the absolute values, the worst tenant perceives a latency of 15 *ms*, which is roughly 1.5 times as much as the best tenant with 9 *ms*. In particular in data centers, where milliseconds count, such latency overhead might lead to unpredictable application performance for some tenants.



(a) Latencies over time. **(b) Heatmap.**

Figure 2: Evaluation results: one run over time and comprehensive presentation of interference in terms of maximum difference among all median values of the latency distribution of tenants. Latency values per tenants are averaged over time intervals of 1 second.

Next, we provide a comprehensive comparison of the latency distributions between tenants among all setups. Our idea is to compare the maximum difference of median values of the latency distributions of tenants for all settings; the maximum difference among all median pairs indicates interference. Simply speaking, the larger the difference, the larger the interference, and, the more unfair is the share in perceived control plane latency in one setting. As shown in Fig. 2b for all settings, the maximum difference increases when the VL faces more tenants and more switches; a setting with 20 tenants and 20 switches shows with 100 *ms* the largest difference, i.e., unfair latency distribution among all control plane connections.

4 CONCLUSION

We make two contributions: (1) we provide *perfbench*, a tool facilitating easier performance studies for SDN networks due to its ability to emulate both controllers and switches even simultaneously; (2) our initial study of control plane performance identifies an interesting problem for future research — detecting and mitigating control plane interference induced by virtualization.

ACKNOWLEDGEMENT

This work has been performed in part in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1) funded by the German BMBF (Project ID 16KIS0473), and in part in the framework of the EU project FlexNets funded by the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

REFERENCES

[1] Arsany Basta, Andreas Blenk, Szymon Dudycz, Arne Ludwig, and Stefan Schmid. 2018. Efficient Loop-Free Rerouting of Multiple SDN Flows. *IEEE/ACM Trans. on Netw. (ToN)* 26, 2 (April 2018), 948–961.

[2] Andreas Blenk, Arsany Basta, Martin Reisslein, and Wolfgang Kellerer. 2016. Survey on Network Virtualization Hypervisors for Software Defined Networking. *IEEE Commun. Surveys & Tutorials* 18, 1 (2016), 655–685.

[3] Michael Dalton et al. 2018. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. In *Proc. USENIX NSDI*. USENIX Association, Renton, WA, 373–387.

[4] Jeffrey C Mogul and Lucian Popa. 2012. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Comput. Commun. Rev.* 42, 5 (2012), 44–48.

[5] perfbench. 2017. https://github.com/tum-lkn/perfbench.

[6] Rob Sherwood et al. 2010. Carving Research Slices out of Your Production Networks with OpenFlow. *ACM SIGCOMM Comput. Commun. Rev.* 40, 1 (Jan. 2010), 129–130.

[7] Allan Vidal, Christian Esteve Rothenberg, and Fábio Luciano Verdi. 2014. The libfluid OpenFlow driver implementation. In *Proc. 32nd Brazilian Symp. Comp. Netw.(SBRC)*. 1029–1036.