The Softwarised Network Data Zoo

Manuel Peuster
Paderborn University
manuel.peuster@uni-paderborn.de

Stefan Schneider
Paderborn University
stefan.schneider@uni-paderborn.de

Holger Karl
Paderborn University
holger.karl@uni-paderborn.de

***Abstract*—More and more management and orchestration approaches for (software) networks are based on machine learning paradigms and solutions. These approaches depend not only on their program code to operate properly, but also require enough input data to train their internal models. However, such training data is barely available for the software networking domain and most presented solutions rely on their own, sometimes not even published, data sets. This makes it hard, or even infeasible, to reproduce and compare many of the existing solutions. As a result, it ultimately slows down the adoption of machine learning approaches in softwarised networks.**

To this end, we introduce the "softwarised network data zoo" (SNDZoo), an open collection of software networking data sets aiming to streamline and ease machine learning research in the software networking domain. We present a general methodology to collect, archive, and publish those data sets for use by other researchers and, as an example, eight initial data sets, focusing on the performance of virtualised network functions.

I. Introduction

The softwarisation of networks is considered as the key enabler for more agile, automated, and autonomous network management concepts, including the use of DevOps paradigms [1]. The two key technologies to achieve this are software-defined networking (SDN) and network function virtualisation (NFV). Both play an important role as part of an emerging trend, called zero touch network & service management (ZSM), which aims to remove any manual steps from network management tasks [2].

To automate network management and to realise ZSM, more and more machine learning (ML) and artificial intelligence (AI)-based network management solutions arise which claim to be able to manage and optimise different aspects of our networks [3]. Many of them focus on automated resource dimensioning for NFV scenarios which try to optimise the amount of resources assigned to each involved virtual network function (VNF). To do so, they predict the upcoming network load as well as the performance a VNF achieves using a given amount of resources [4], [5].

But ML-based solutions are always data-driven and do not only depend on programming code, which is a huge difference to legacy management and automation approaches. This means that ML approaches can only work efficiently if enough data is available to train and test the involved models, before they are put into production. Even if data is available in some custom environments, e.g., in form of volatile monitoring metrics, we are still missing a publicly available collection of open data sets that can be used to evaluate and compare different ML solutions and algorithms with each other. Such open data sets are a common tool in other communities, like image recognition [6] or natural language processing [7], and accelerated the adoption of ML solutions in those domains.

We argue that the software networking community also needs open data sets to simplify and streamline ML research and to improve the reproducibility of new ideas. To this end, we introduce the "softwarised network data zoo" (SNDZoo)—an open repository to collect, host, and share software networking data sets. The SNDZoo is, to the best of our knowledge, the first effort to build such a central repository for softwarised network data. It specifically focuses on NFV and SDN performance data sets collected through performance measurements of real-world network setups. These data sets go beyond existing collections of open networking data sets such as topology data sets or traffic traces [8], [9].

Our contributions are two-fold. After discussing related work in Section II, we first introduce a generic *methodology and workflow* to collect data sets for the SNDZoo, using our open-source NFV benchmarking platform [10], in Section III. Second, we present *eight data sets*, containing millions of performance measurements collected from different real-world VNFs from the security, web, and 5G vertical (Internet of things) areas, in Section IV. We make all data sets available as part of the SNDZoo project [11] for use by other researchers.

II. Related Work

As in many other domains, ML recently found its way into the information and communications technology (ICT) sector and networking domain [3], [12], [13]. The use of ML is especially appealing for network management and optimisation use cases in softwarised networks. It is specifically well suited for NFV/SDN scenarios, which shall be highly automated to allow zero-touch network operations following DevOps methods [1], [2]. Existing work focuses on, e.g., learning and predicting of service metrics, such as response time and frame rate [14], [15], scaling and resource dimensioning [4] as well as placement decisions [5]. But all of this work relies on custom data sets which might not even be publicly available. The available public data sets, such as [14], are however not available at a common repository and thus are often hard to find. Our work improves this situation by offering a common repository to host and share data sets focusing on performance measurements of softwarised networks as well as the involved platforms and components. This also complements and goes beyond existing collections of open network data sets mainly focusing on network topology graphs, like [8], [9], [16].

978-3-903176-24-9 © 2019 IFIP

To collect the presented data sets, automated solutions to benchmark and profile NFV and SDN scenarios, including our own work [10], [17], can be used [18]–[21]. Those benchmarking approaches have not only been proposed by academia but have also been in the focus of standardisation bodies [22], [23]. They are complementary to the work presented in this paper. In fact, we make use of these solutions to collect the initial data sets available in the SNDZoo [11].

III. Methodology & Workflow

Collecting data sets from softwarised network scenarios is more than performing a handful of manual measurements on a testbed running in a lab. In fact, manual measurements should be avoided where possible to be able to (i) quickly and objectively reproduce the measurements, (ii) run a given measurement in a new environment, e.g., outside the lab, and (iii) make use of the fact that software-based networking scenarios can be automatically deployed, allowing to completely remove all manual steps from the process. On top of that, software-based networks usually offer a much higher degree of configuration freedom, e.g., in terms of virtual resources assigned to a VNF, resulting in many different configurations for which measurements can be and need to be performed [24].

We introduce such a fully-automated data collection methodology in Figure 1 using an NFV scenario as example. A data collection setup consists of two main components. First, an NFV benchmarking framework that is responsible to control, manage, and automate the measurement and collection process. Second, one or multiple NFV platforms, consisting of management and orchestration (MANO) layer and NFV infrastructure. They are used to deploy and execute the experiment setups, including the deployment and execution of the system under test (SUT). This concept is independent of the technical realisation of the different components. It is, for example, possible to use different benchmarking frameworks, such as tng-bench [10], Gym [18], or NFV Inspector [21], or multiple NFV platforms, such as vim-emu [25], [26], OSM [27], or SONATA-NFV [28], to perform measurements in different environments.

Using the setup shown in Figure 1, the following end-to-end workflow is used to collect a full data set, containing measurements for many different configurations of a given SUT. First, the measurement experiment is defined using the description approach offered by the used benchmarking framework (1), e.g., based on our current standadisation efforts [29]. This description contains, e.g., a list of different configurations that should be tested as well as a definition of the probes that should be used. Probes are VNFs that can be deployed together with and connected to the SUT and are used to stimulate the SUT during the experiments. Our system is not fixed to a particular probe implementation and custom probes are possible, e.g., synthetic traffic generators or replayed traffic traces. The final description as well as the SUT (e.g., given as an ETSI SOL004 VNF package) is then used as input for the framework. The framework then on-boards and deploys the first configuration to be tested on the

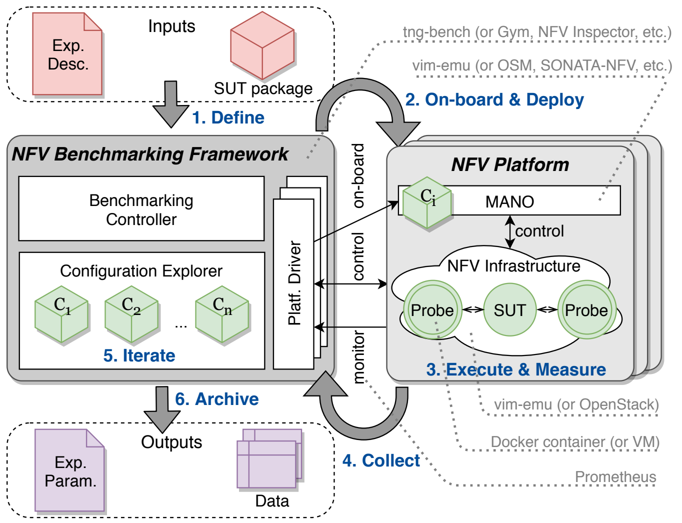

Fig. 1: Example data collection setup and workflow. The dotted notes point to possible implementation options.

NFV platform by triggering the MANO system (2). Once the SUT and the defined probes are instantiated and configured, the framework triggers the execution of the experiment, i.e., activating the probes and measuring the performance achieved by the SUT (3). When the execution is done (e.g., after a fixed time limit), the SUT and the probes are terminated and the results are collected by the framework (4), before the next configuration is selected (5) and a new measurement cycle is started (2). Finally, after all configurations have been successfully executed, the framework combines all collected results with the configuration parameters to label the data and archive it in a table-based format (6).

We implement the presented workflow using our own NFV benchmarking framework, called *tng-bench*, which was initially presented in [10] and is now available as actively-developed open-source project [30]. We use vim-meu [25] as NFV platform to execute container-based SUTs and probes. There are two kinds of metrics collected by tng-bench. First, the experiment metrics that are collected from the probes as well as the SUT at the end of an experiment, e.g., total number of processed packets. Those metrics are captured by collecting log files from the involved containers before termination. Second, we collect time series metrics using the Prometheus time series database [31] controlled by tng-bench. Prometheus periodically fetches all metrics, including the resource usage of the involved containers (using cAdvisor [32]) as well as SUT-specific metrics as we detail in the following section.

IV. Collecting, Archiving, and Publishing the First Data Sets

We collected eight initial data sets, focusing on the performance of real-world VNFs under different configurations, to kick-off the SNDZoo project and to test the presented methodology, workflow, and tools. All initial data sets are automatically collected and can be reproduced using our benchmarking tool. We want to highlight that the scope of the SNDZoo is not limited to performance data sets of single VNFs. We plan to extend our efforts and also include measurements of more complex network services, MANO performance numbers, impact of different management approaches to the performance of real world NFV systems, or even long-term monitoring data into the project.

A. Experiment Setup

To collect the data sets, we picked eight VNFs from three different categories as shown in Table I: Security (IDS systems), web (load balancers, proxies), and IoT (MQTT brokers). This way, we not only have VNFs that transparently forward the traffic while passively analysing it (IDS systems), but also active VNFs that can modify the traffic (proxies). We also have scenarios (MQTT broker) that can be considered as examples for 5G vertical use cases, such as IoT, smart manufacturing, or industry 4.0 [33].

In the first category (`SEC01-SEC03`), we benchmark three IDS VNFs, namely Suricata 4.0 [34], Snort 2.9 [35], and Snort 3.0 [35]. Each IDS is configured as transparent layer 2 bridge and passively monitors the incoming traffic. To stimulate the VNFs we use two publicly available traffic traces with small and big flows that are continuously replayed at maximum speed [36]. During the benchmarking experiment, the VNFs are configured with different IDS rule sets, taken from [37], and with different resource assignments, i.e., CPU time (10 % to 100 %) and memory (256 MB and 1024 MB). As a result, 80 different configurations are tested[1] and each configuration is repeated 20 times, resulting in 1,600 experiment runs, each testing a single configuration.

The second category (`WEB01-WEB03`) represents web scenarios in which we test an Nginx 1.10.3 [38] and a HAProxy 1.6.3 [39] load balancer VNF as well as a Squid 3.5.12 [40] proxy. The VNFs are placed between a source probe (user requests generated by Apache Bench [41]) and a target probe (web server running Apache 2.0 [41]). As shown in Table I, 80 different configurations are executed, including small and large requests and different resource assignments, i.e., CPU time (10 % to 100 %) and memory (64 MB, 128 MB, 256 MB, and 512 MB).

In the third category (`IOT01-IOT02`), the MQTT brokers Mosquitto 1.6.2 [42] and Emqx 3.1.0 [43] are tested using Malaria [44], an MQTT load generator. The broker is placed between two probes running Malaria instances, one acting as publisher and the other acting as subscriber, allowing us to measure the end-to-end delay of MQTT messages. Besides different resource assignments for the broker VNF, Malaria is executed with different configurations, e.g., message sizes between 10 and 1000 bytes as well as two different MQTT QoS levels (1 and 2). Please note that the full details of all used configurations, versions, and workloads are published along with the data sets [11].

[1]There are only 40 configurations in the case of Snort 3.0 because of the smaller rule set available for this version of Snort.

B. Data Collection

To collect the presented data sets we used the setup presented in Section III using vim-emu [25] as NFV platform, which is executed on a machine with Intel(R) Xeon(R) W-2145 CPU at 3.70 GHz CPU, 32 GB of memory, running Linux kernel 4.4.0-142-generic[2]. Vim-emu allows to deploy and control NFV scenarios on a single physical machine using Docker containers. In our experiments, the tested VNFs as well as the probes used to stimulate them are deployed as Docker containers, each of them always pinned to a single physical CPU core to achieve isolation between VNFs and probes [10].

We collect a large number of different experiment metrics for each tested configuration as well as a large number of time series metrics during experiment execution. Those metrics are then labeled with the used configurations. More specifically, we collect 268 to 281 experiment metrics after each experiment and between 43 and 593 time series metrics during each experiment. This results in up to 474,400 collected time series records in data set `SEC03`[3], as shown in Table I. Each of these records contains about 60 data points resulting from a collection frequency of 0.5 Hz and an experiment runtime of 120 s per configuration. Those numbers highly depend on the involved VNFs and the number of metrics they expose. The used Snort 3.0 VNF, for example, exposes more than 400 metrics that can be collected, e.g., packet counters for different protocol types.

The use of tng-bench as experiment automation framework allows to reproduce all presented experiments. To do so, nothing more is required than two Linux machines on which tng-bench and vim-emu are installed. All involved VNFs can be downloaded from the SNDZoo repositories as pre-defined Docker containers [11] and used to re-run the experiments. However, the absolute performance numbers in those data sets obviously depend on the underlying hardware and will differ for new measurements performed in different environments. But, generic aspects like trends that can be identified between different VNF configurations will still be visible [17].

C. Resulting Data Sets

The presented data sets contain between 4.6 and 28.7 million data points and are, for example, usable as training and testing data sets for different kinds of prediction or optimisation algorithms in the NFV domain. The collection process of each data set took between 35.8 h and 115.8 h, as shown in Table I. Even though the `SEC03` data set has the shortest runtime, it contains the most data points. The reason for this is that the tested VNF, Snort 3.0, exposes more VNF-specific metrics than any other tested VNF. The measurements to collect `IOT02` took more time than the others because the used VNF (Emqx) takes much longer to start up.

Figure 2 visualises a small subset of our data sets to give the reader a brief example of the published data and to show

[2]The full hardware and software specifications of the used testbed are available as part of the SNDZoo repositories.

[3]Considering configurations, repetitions, and collected metrics results in: $40 \cdot 20 \cdot 593 = 474{,}400$ time series records.

TABLE I: Overview of the eight VNF benchmarking data sets initially published in the SNDZoo [11]

Name	Category	Class	VNF	Probe/Stimuli	Tested Configu-rations	Repetitions	Experiment Metrics	Time Series Metrics	Total Exp. Runtime	Total data points
SEC01	Security	IDS Systems	Suricata [34]	Traces [36]	80	20	280	157	71.1 h	15.5M
SEC02			Snort 2.9 [35]	Traces [36]	80	20	280	169	72.5 h	16.7M
SEC03			Snort 3.0 [35]	Traces [36]	40	20	281	593	35.8 h	28.7M
WEB01	Web	Load balancers	Nginx [38]	AB/Apache [41]	80	20	268	43	70.4 h	4.6M
WEB02			HAProxy [39]	AB/Apache [41]	80	20	268	43	70.2 h	4.6M
WEB03		Proxys	Squid [40]	AB/Apache [41]	80	20	268	43	70.5 h	4.6M
IOT01	IoT	MQTT Broker	Mosquitto [42]	Malaria [44]	80	20	275	90	71.6 h	9.1M
IOT02			Emqx [43]	Malaria [44]	80	20	275	109	115.8 h	10.9M

how researchers can make use of it. The figure shows the comparison of two metrics (requests/s and request times) of the WEB01-WEB03 data sets for different CPU and memory configurations of the three SUTs (nginx, haproxy, and squid). Each configuration is represented by 20 points (measurement repetitions). The lines indicate polynomial regressions (degree=3) to demonstrate how the data can be used to train performance models mapping different SUT configurations to performance metrics. Those models could, e.g., be used for scaling and placement optimisation algorithms [45].

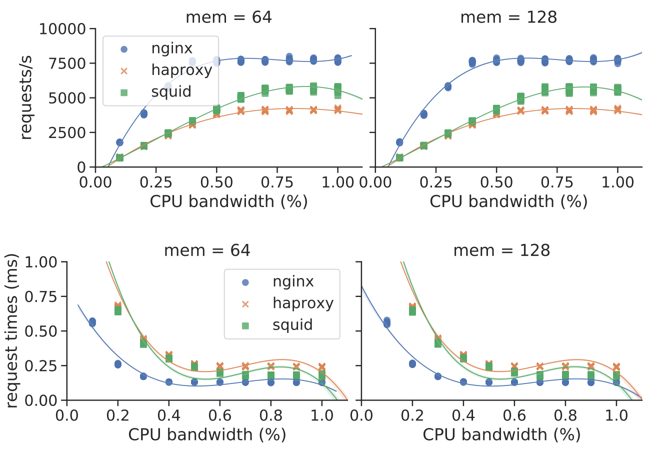

Fig. 2: Example data set visualisations

D. Publishing the Data Sets

We version all experiment configurations that are used to collect the presented data sets using Git so that we can reproduce old experiments even if they are updated or improved over time. This also allows us to use GitHub as a hosting platform to publish and share the contents of the SNDZoo. However, keeping large data sets, containing multiple files with gigabytes of data, within a Git repository is bad practice and results in poor performance. To solve this, we make use of a recently introduced project called Data Version Control (DVC) [46]. DVC focuses specifically on versioning ML data sets and allows to store and version large files on external storage solutions such as Amazon S3 while referencing them from a Git repository. Users can then access and download a specific data set by simply running two commands (`git clone` and `dvc pull`) on their machine.

Each data set hosted in the SNDZoo is located in its own Git/DVC repository with a connected Amazon S3 bucket holding the data files (a total of 7.3 GB). Those repositories not only contain the configurations and resulting data sets but also meta data like the hardware and software specifications of the machines on which the measurements have been performed. Further, they contain license information as well as all raw measurements, time series, and logs produced by tng-bench. Besides the data set repositories, SNDZoo also provides repositories containing the sources and descriptions of used VNFs and services. All published data sets are indexed on and linked from the SNDZoo website [11]. They are published under creative commons CC-BY-SA 4.0 license.

V. Conclusion

We introduced the SNDZoo project as the first and (at the time of writing) largest open collection of NFV/SDN performance data sets. The SNDZoo project aims to support the adoption of ML/AI in the software networking community by enabling researchers to work with common data sets simplifying comparisons and reproduction of results. Along with this paper, we publish eight initial data sets, containing performance measurements collected from a set of real-world security, web, and IoT VNFs.

We plan to continue our efforts and add new data sets over time. Our broader vision for the SNDZoo is to have a large collection of different data sets for a wide variety of use cases, scenarios, and deployments. Not only focusing on NFV and SDN performance measurements but also on measurements of MANO system and management performance. This project, however, depends on community contributions and we invite all community members to participate and use SNDZoo as a platform to host and share their data sets. More details about the contribution process are available online [11].

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO), and the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

References

[1] H. Karl, S. Dräxler, M. Peuster, A. Galis, M. Bredel, A. Ramos, J. Martrat, M. S. Siddiqui, S. van Rossem, W. Tavernier *et al.*, "DevOps for network function virtualisation: an architectural approach," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1206–1215, 2016.

[2] J. Miao, G. He, X. Pei, K. Martiny, M. Klotz, A. Khan, G. K. nad Ryosuke Kurebayashi, Y. Goto, S. Manning, F. Weaver, and D. Lopez. (2017, Dec.) Zero-touch Network and Service Management. [Online]. Available: https://portal.etsi.org/TBSiteMap/ZSM/OperatorWhitePaper

[3] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, March 2018.

[4] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "Topology-aware prediction of virtual network function resource requirements," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 106–120, March 2017.

[5] J. Sun, G. Huang, G. Sun, H. Yu, A. K. Sangaiah, and V. Chang, "A q-learning-based approach for deploying dynamic service function chains," *Symmetry*, vol. 10, no. 11, 2018. [Online]. Available: http://www.mdpi.com/2073-8994/10/11/646

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[7] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150.

[8] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007.

[9] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[10] M. Peuster and H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[11] M. Peuster, S. Schneider, and H. Karl. (2019, Apr.) Softwarised Network Data Zoo. [Online]. Available: https://sndzoo.github.io

[12] S. Xu, Y. Qian, and R. Q. Hu, "Data-driven network intelligence for anomaly detection," *IEEE Network*, vol. 33, no. 3, pp. 88–95, May 2019.

[13] M. Usama, J. Qadir, A. Raza, H. Arif, K. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019.

[14] R. Stadler, R. Pasquini, and V. Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 672–698, Oct 2017. [Online]. Available: https://doi.org/10.1007/s10922-017-9426-z

[15] F. S. Samani and R. Stadler, "Predicting distributions of service metrics using neural networks," in *2018 14th International Conference on Network and Service Management (CNSM)*, Nov 2018, pp. 45–53.

[16] J. Leskovec and A. Krevl. (2014, Jun.) SNAP Datasets: Stanford large network dataset collection. [Online]. Available: http://snap.stanford.edu/data

[17] M. Peuster and H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management," in *5th European Workshop on Software Defined Networks (EWSDN'16)*. IEEE, 2016.

[18] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take your vnf to the gym: A testing framework for automated nfv performance benchmarking," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 110–117, 2017.

[19] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "NFV-VITAL: A Framework for Characterizing the Performance of Virtual Network Functions," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 93–99.
York, NY, USA: ACM, 2018, pp. 14:1–14:13. [Online]. Available: http://doi.acm.org/10.1145/3185467.3185495

[20] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for vnf and service chain analysis in software-defined nfv," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New

[21] M. G. Khan, S. Bastani, J. Taheri, A. Kassler, and S. Deng, "Nfv-inspector: A systematic approach to profile and analyze virtual network functions," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 2018, pp. 1–7.

[22] R. V. Rosa, C. E. Rothenberg, M. Peuster, and H. Karl, "Methodology for VNF Benchmarking Automation," IETF, Internet-Draft draft-rosa-bmwg-vnfbench-02, Jul. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-rosa-bmwg-vnfbench-02

[23] ETSI. (2018) TST009v3.1.1: Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI. [Online]. Available: https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20009v3.1.1%20-%20GS%20-%20NFVI_Benchmarks.pdf

[24] M. Peuster and H. Karl, "Understand Your Chains and Keep Your Deadlines: Introducing Time-constrained Profiling for NFV," in *2018 14th International Conference on Network and Service Management (CNSM)*, Nov 2018, pp. 240–246.

[25] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-ready Network Services in Multi-PoP Environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, pp. 148–153.

[26] ETSI OSM, "OSM vim-emu documentation," https://goo.gl/XZNLVw.

[27] ——. (2016) Open Source MANO: Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV. [Online]. Available: https://osm.etsi.org

[28] SONATA project consortium. (2015) SONATA-NFV. [Online]. Available: http://sonata-nfv.eu

[29] R. Rosa and C. Rothenberg, "VNF Benchmarking Methodology," IETF Internet-Draft https://tools.ietf.org/id/draft-rosa-bmwg-vnfbench-01.html, UNICMAP, Internet-Draft, 2018. [Online]. Available: https://tools.ietf.org/id/draft-rosa-bmwg-vnfbench-01.html

[30] 5GTANGO project consortium. (2018) 5GTANGO SDK: tng-sdk-benchmark. [Online]. Available: https://github.com/sonata-nfv/tng-sdk-benchmark

[31] Linux Foundation. (2019) Prometheus Time Series Database. [Online]. Available: https://prometheus.io/

[32] Google Inc. (2019) Google cAdvisor. [Online]. Available: https://github.com/google/cadvisor

[33] S. Schneider, M. Peuster, D. Behn, M. Müller, P.-B. Bök, and H. Karl, "Putting 5G into Production: Realizing a Smart Manufacturing Vertical Scenario," in *2019 IEEE European Conference on Networks and Communications (EuCNC)*, 2019.

[34] OISFoundation. (2019) Suricata: Open Source IDS / IPS / NSM engine. [Online]. Available: https://suricata-ids.org/

[35] Cisco. (2016) Snort IDS/IPS. [Online]. Available: http://www.snort.org

[36] F. Klassen and AppNeta. (2019) Tcpreplay: Sample Captures. [Online]. Available: http://tcpreplay.appneta.com/wiki/captures.html

[37] ET Labs. (2019) Emerging Threats Open Ruleset. [Online]. Available: https://rules.emergingthreats.net/

[38] NGINX Inc. (2019) NGINX: High Performance Load Balancer, Web Server, Reverse Proxy. [Online]. Available: https://www.nginx.com/

[39] HAProxy Project. (2019) HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer. [Online]. Available: http://www.haproxy.org/

[40] Squid Project. (2019) Squid: Optimising Web Delivery. [Online]. Available: http://www.squid-cache.org

[41] The Apache Software Foundation. (2019) Apache Project. [Online]. Available: https://httpd.apache.org/

[42] Eclipse Foundation. (2019) Eclipse Mosquitto: An open source MQTT broker. [Online]. Available: https://mosquitto.org/

[43] EMQ Technologies Co., Ltd. (2019) EMQ: Scalable and Realtime MQTT Messaging for IoT in 5G Era. [Online]. Available: https://www.emqx.io/

[44] Malaria Project. (2019) Malaria: Attacking MQTT systems with Mosquittos. [Online]. Available: https://github.com/etactica/mqtt-malaria

[45] S. Dräxler, H. Karl, and Z. Á. Mann, "Jasper: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 946–960, Sep. 2018.

[46] DVC Project. (2019) DVC: Open-source Version Control System for Machine Learning Projects. [Online]. Available: https://dvc.org/