Emulation-based Smoke Testing of NFV Orchestrators in Large Multi-PoP Environments

Manuel Peuster*, Michael Marchetti†, Gerardo García de Blas‡ and Holger Karl*
*Paderborn University: {manuel.peuster, holger.karl}@upb.de
† Sandvine: mmarchetti@sandvine.com
‡ Telefónica Investigación y Desarrollo: gerardo.garciadeblas@telefonica.com

***Abstract*—Management and orchestration (MANO) systems are the key components of future large-scale NFV environments. They will manage resources of hundreds or even thousands of NFV infrastructure installations, so called points of presence (PoP). Such scenarios need to be automatically tested during the development phase of a MANO system. This task becomes very challenging because large-scale NFV testbeds are hard to maintain, too expensive, or simply not available.**

In this paper, we present a multi-PoP NFV infrastructure emulation platform that enables automated, large-scale testing of MANO stacks. We show that our platform can easily emulate hundreds of PoPs on a single physical machine and reduces the setup time of a test PoP by a factor of 232× compared to a DevStack-based test PoP installation. Further, we present a case study in which we test ETSI's Open Source MANO (OSM) against our proposed system to gain insights about OSM's behaviour in large-scale NFV deployments.

I. Introduction

Network function virtualization (NFV) is expected to be one of the key enablers for highly agile network service deployments in future 5G networks. The key component of each NFV system is its management and orchestration (MANO) stack that controls the deployment and configuration of individual virtualized network functions (VNF) as well as complex network services (NS). Several of these MANO stacks are currently developed, some as commercial products, others as open source community or research projects, like SONATA [1], ONAP [2], or Open Source MANO (OSM) [3].

Those MANO systems are complex software systems that are built to control multiple NFV infrastructures (NFVI) provided by many spatially-distributed cloud data centers through so-called virtual infrastructure managers (VIM), e.g., OpenStack installations [4]. Such deployments are also called multi-point-of-presence (multi-PoP) environments, where each PoP is assumed to provide some NFVI resources as well as a VIM that offers interfaces to request the available resources, e.g., start a VNF. This scenario leads to a big challenge for the development of MANO systems, which obviously need to be tested against such large-scale environments. The problem here is that large multi-PoP environments are costly, hard to set up, and usually are just not available to MANO developers. Even if they are available, it is often too expensive to use them in automated test pipelines for continuous integration (CI), which would occupy resources whenever a developer submits code and triggers a new test run.

In this paper, we present a solution for this: An emulation-based test platform that can emulate multiple NFVI environments on a single machine, enabling automated tests of MANO systems in large multi-PoP scenarios. The presented solution is inspired by a concept called *smoke testing* [5]. The *smoke testing* concept focuses on testing only the main functionality of a complex system and skips unimportant details to reduce test times. Our platform does exactly this by re-implementing a subset of the OpenStack APIs, the de-facto standard for VIM interfaces, today. A MANO system can then use these APIs to deploy container-based test services on the emulated NFVI PoPs and thus verify the MANO system in end-to-end scenarios.

The contributions of this paper are as follows: First, we describe how we extended the emulation platform, presented in our previous work [6], to act as an emulated test environment for MANO systems and how we integrate it with an automated test pipeline in Sec. III-A. Second, we analyze the scalability of our platform to show that it can easily emulate hundreds of PoPs on a single physical machine or VM in Sec. III-C. Finally, Sec. IV presents a case study in which we tested OSM [3] against our platform and discovered some interesting insights and bugs that would not have been found with existing, lab-scale NFVI testbeds offering only a handful of PoPs.

II. Related Work

NFV development support and especially automated testing in the NFV domain is still a novel research direction with a limited amount of existing solutions. Some recent work focuses on end-to-end testing in 5G networks [7] or the verification and validation of network services [8]. But none of them explicitly considers the need of testing the core part of NFV deployments: The MANO system. In the software community, smoke testing has already been established since several years, providing the ability to quickly integrate new versions of different software components [5], which is what our solution introduces for NFV MANO systems.

One way to setup end-to-end smoke tests for a MANO system is to use either local or remote testbed installations. The problem with local installations, like [9], are their resource limits which prevent large-scale test cases, e.g., with a high number of PoPs. Remote testbeds, like [10]–[12], may offer the required NFV infrastructure and interfaces, but their main focus is the development and evaluation of network services. In addition, they are shared between many users which means they may not always be available to quickly execute automated tests on them. In general, these testbed solutions are complementary to our presented approach and should be used for final, manually-deployed integration tests rather than for automated smoke testing.

Another option for automated smoke tests is using locally available network emulation approaches, like Mininet [13], CORE [14], or VLSP [15]. Unfortunately, these solutions focus on prototyping and evaluation of new protocols or network management paradigms rather than on interactions with production-ready MANO solutions. None of these solutions offers de-facto standard VIM northbound interfaces for easy MANO system integration, like our solution does with its OpenStack-like interfaces. Even if VLSP focuses on MANO-like experiments in the NFV domain, it lacks the ability to execute real-world VNF software, which is possible in our platform that uses lightweight container solutions to run VNFs in an emulated environment.

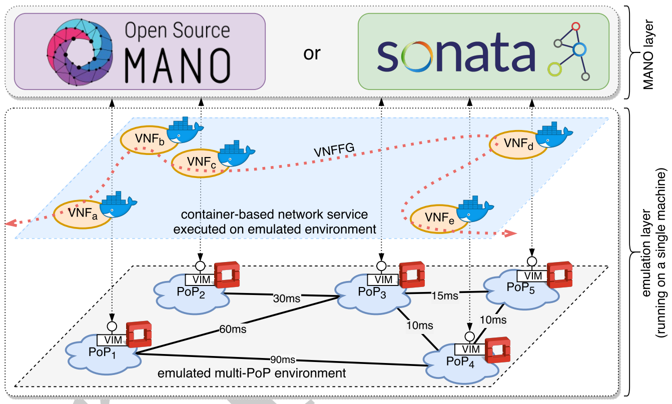
III. An NFV multi-PoP Test Platform

The presented platform is based on our previous work on an emulation-based rapid prototyping platform, called *MeDICINE* [6], which allows to emulate realistic multi-PoP topologies. This work extends the emulated PoPs of the *MeDICINE* platform with OpenStack-like northbound APIs to allow their integration with real-world MANO systems, a concept that was initially demonstrated in [16]. In this paper, we go beyond this initial integration and use our extended emulation platform as part of an automated test pipeline to speed up tests of complex, real-world MANO systems and analyze some of their scaling capabilities.

A. Architecture & Design

Our emulation platform consists of three main components. First, the network emulation part, which is based on Containernet [17], a Mininet extension [13]. It allows to execute network functions inside Docker containers that are connected to arbitrary, user-defined network topologies [6]. Second, the *VIM emulation* part, which creates an abstraction layer for the network emulation and allows a user to define arbitrary topologies with emulated NFVI PoPs, each representing a single VIM endpoint. This allows the emulation platform to emulate realistic, distributed NFVI deployments, e.g., by adding artificial delays to the links between the PoPs. The abstraction layer allows to deploy single VNFs, in form of Docker containers, inside each of the emulated PoPs. The third part, which is one of the main contributions of this paper, are additional APIs on top of the emulation platform. These APIs mimic the original OpenStack APIs for each of the emulated PoPs and translates OpenStack requests, e.g., `openstack compute start`, into requests that are then executed by the emulation platform, e.g., start a Docker-based VNF in one of the emulated PoPs.

Figure 1 shows a usage scenario in which our emulation platform (bottom layer) emulates five interconnected PoPs, each offering its own OpenStack-like northbound API. This emulated infrastructure can be controlled by any real-world MANO system that is able to use OpenStack, e.g., OSM [3] or SONATA [1] (top layer). The MANO system is used to instantiate a complex, distributed network service, consisting of five VNFs, on top of the emulated infrastructure (middle layer). With this setup, the emulated infrastructure and the instantiated services look like a real-world multi-PoP NFVI deployment from the perspective of the MANO system.



Open Source MANO
or
sonata
MANO layer
VNF$_b$
VNF$_c$
VNFFG
VNF$_d$
VNF$_a$
container-based network service executed on emulated environment
VNF$_e$
VIM
PoP$_2$
30ms
PoP$_3$
15ms
PoP$_5$
10ms
10ms
60ms
PoP$_1$
90ms
PoP$_4$
emulated multi-PoP environment
emulation layer (running on a single machine)

Fig. 1: A multi-PoP topology with five emulated, OpenStack-like NFVIs running on a single physical machine (bottom) and five Docker-based VNFs running on the emulated infrastructure (middle), all controlled by a real-world MANO system (top).

B. Emulation-based Smoke Testing

Our emulation platform can be used to rapidly prototype network services or to experiment with different MANO solutions on a local machine [6] as shown in Fig. 1. But it can also be used for *automated testing of MANO systems*. As mentioned earlier, integration tests for NFV MANO systems are complicated to set up and usually require a huge amount of resources, especially if tests against multi-PoP infrastructures should be performed. Our solution resolves these issues by testing the MANO system against the emulated multi-PoP infrastructure instead of using real, multi-PoP NFVI deployments. We call this approach *emulation-based smoke testing*.

Fig. 2 shows the proposed testing setup in which a test controller, e.g., Jenkins[1] or a simple shell script, automatically sets up our emulation platform with a pre-defined multi-PoP topology (1). Once this is done, it configures the MANO system to be tested, e.g., OSM, connects it to the VIM interfaces of the emulated PoPs, and finally triggers some test requests against the MANO's northbound interface, e.g., deploying a test service (2). The test controller can then check if the resulting deployments on the emulated infrastructure are correct (3). Once all tests are done, the test controller destroys the emulated infrastructure by stopping the emulation platform

[1]Jenkins CI: https://jenkins.io

and can start a new emulation instance, e.g., with a different multi-PoP topology, for further tests.

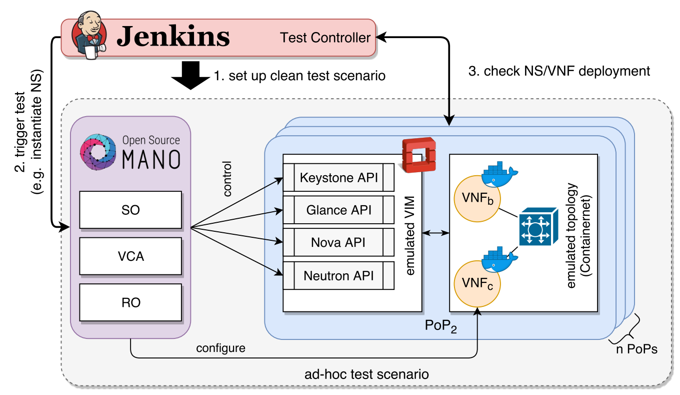


Fig. 2: A automated testing setup for a MANO system, e.g., OSM. The test controller automatically sets up the emulated infrastructure (multiple PoPs) and tests the MANO system against this fresh infrastructure.

Using an emulation-based test infrastructure provides some major benefits when compared to a real multi-PoP OpenStack deployment. First, the state of the emulated VIM and NFVI is volatile, which ensures that tests are always executed in a clean environment, e.g., there are no zombie VMs left in a PoP. Second, the setup of the emulation platform is much quicker and the required resources are far less than for a full-featured VIM, e.g., an OpenStack installation. More importantly, the emulation platform can be executed on a single machine (physical or VM), making it a much better fit for existing test pipelines, e.g., a Jenkins installation. Third, the emulated infrastructure can be easily scaled to hundreds of PoPs whereas an automated, interconnected setup of hundreds of OpenStack installations is very challenging.

To get a first idea about the setup time savings that can be expected from emulated PoPs, we compared the setup times of our emulation platform configured to emulate a single OpenStack-like PoP with the setup times of a single-node OpenStack DevStack [4] installation, which can be considered as the most simple way to install a fully-featured OpenStack in a PoP. We executed both setup procedures 10 times on a single physical machine with Intel(R) Core(TM) i5-4690 CPU @ 3.50 GHz and 16 GB memory and found a mean setup time for a single emulated PoP of 2.48 s compared to a mean setup time of 576.42 s for a fresh DevStack installation, which is more than 232 times slower. This comparison makes sense since we want to ensure that we always test against a clean environment and thus a fresh installation of DevStack would be always required. When considering multi-PoP deployments with a large number of PoPs, the benefits of our emulation platform become even more interesting, as discussed in Sec. III-C.

As expected, there are also a couple of limitations when using emulation-based infrastructure for testing. First, not all features of the original OpenStack APIs are supported by our platform. In the current implementation, we focused on the API endpoints required to let typical MANO solutions, like OSM, believe that it talks to a real OpenStack installation, namely the *Keystone*, *Nova*, *Glance*, and *Neutron* endpoints. However, new endpoints can easily be added to the emulation framework. Second, the emulated infrastructure is only able to deploy VNFs based on containers instead of full-blown VMs. This limitation is required to keep the the emulation lightweight. Third, the total available resources of the emulated infrastructure is limited since it is executed on a single machine. However, the lightweight emulation design still allows to emulate hundreds of PoPs as shown in Sec. III-C. These limitations must be kept in mind when using our platform in a testing pipeline. In general, our *emulation-based smoke tests* should not be considered as a full replacement of a final integration test against a real multi-PoP environment but as a much faster, intermediate testing stage that can easily be executed for each new commit to the MANO system's code base—something that is certainly not feasible with existing setups based on real-world PoP installations.

C. Multi-PoP Scalability Testing

In addition to functionality-focused integration tests between MANO system and emulated VIMs, our platform also enables *advanced scalability testing* with a very large number of PoPs and/or a high number of deployed services. These kinds of tests are often infeasible with real-world deployments since globally distributed test networks with tens, hundreds, or even thousands of PoPs are usually not available. Further, tests with hundreds of deployed service instances require a huge amount of resources which are too expensive to just use them in an automated test pipeline. Nevertheless, users of production-ready, carrier-grade MANO systems, like OSM, expect that these systems scale well with the number of attached PoPs and the number of deployed services. To solve this, our emulation-based approach is a perfect fit since it is able to emulate many PoPs and allows to deploy many lightweight services on the emulated infrastructure. All this can be done on a single machine whereas similar DevStack-based testbed installations would require hundreds of machines.

To quantify the scaling abilities of our emulation platform, we did a set of experiments to study its behaviour when topologies with many PoPs are emulated or when hundreds of service instances are deployed on the emulated infrastructure. All experiments have again been executed on a single physical machine with Intel(R) Core(TM) i5-4690 CPU @ 3.50 GHz and 16 GB memory and have been repeated 10 times. In the first experiment, we analyzed the startup and configuration time of the emulation platform for different numbers of PoPs and different PoP interconnection patterns. Fig. 3 shows the setup time breakdown for up to 100 PoPs for three topologies. It shows how much time is used by which of the four phases of the emulation setup procedure: *Initialization*, *PoP setup*, *link setup*, and *emulation start*. The *linear* topology connects all PoPs into a long chain, the *star* topology connects all PoPs to a single central PoP, and the *mesh* topology creates a link between each pair of PoPs in the emulation. All error bars in this paper show standard deviations. The results show that with *linear* and *star* topologies, 100 PoPs can be set up in about 40 s, which is a huge improvement when compared to 100 DevStack installations (see Sec. III-B). In the *mesh* topology case, the setup takes longer which is caused by the much higher number of links that need to be established between each pair of PoPs accounting for the connection setup and emulation start phases.

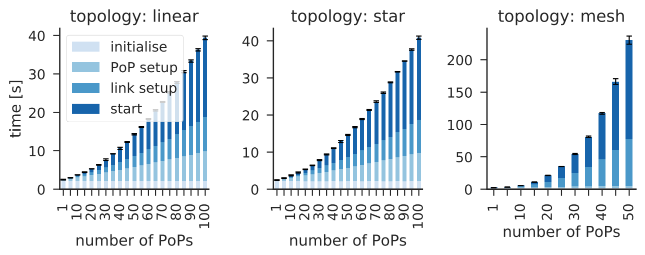
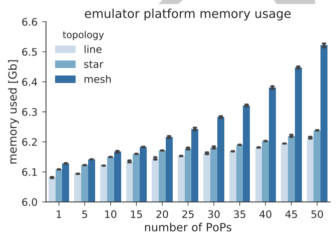


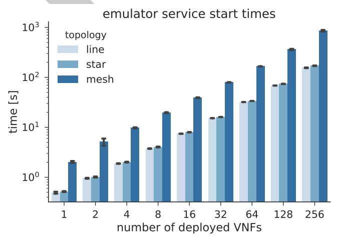
Fig. 3: Set up time breakdown of artificial topologies

We also analyzed the memory consumption for these three scenarios (Fig. 4a). The figure shows the total memory used by the test machine which increases proportionally to the number of PoPs and links in the topology. In general, not more than 6.6 Gb of memory are used, which shows that our emulation platform can easily be executed in existing test nodes or even on a developer's laptop.

Finally, we studied the time required to deploy a large number of VNFs on top of the emulated infrastructure. We again used our *liner*, *star*, and *mesh* topologies with 50 PoPs and deployed up to 256 VNFs on the emulated PoPs (randomly placed). The used VNFs are based on the default Docker `ubuntu:trusty` images and do not run any additional software, since we are only interested in the bare instantiation times. Fig. 4b shows that the instantiation times linearly scale with the number of VNFs and that the instantiation process takes longer in larger topologies. It can be seen that with our platform hundreds of VNFs can be quickly deployed on a single machine which enables fast tests of large deployment scenarios.



(a) Memory usage



(b) Instantiation times

Fig. 4: Memory usage of topologies with up to 50 PoPs and VNF instantiation times for up to 256 VNFs.

IV. Case Study: Testing Open Source MANO (OSM)

We picked OSM [3] as it is one of the most prominent open source MANO solutions and performed a case study to verify the usefulness of the described approach. The setup for the study was the same as described in Fig. 2 but we used a scripted test controller that automatically performs a series of experiments and collects additional data. Besides the general functionality of the *VIM attach* and *network service deployment* procedures, we investigated the behaviour of OSM when it has to interact with large multi-PoP deployments and a high number of instantiated network services. To be more realistic, we used a set of real-world topologies with different sizes that are taken from the *Internet Topology Zoo (ITZ)* library [18]. In our case study, each node of a given topology is turned into a single PoP emulating an OpenStack VIM. These are test cases which are not covered by existing NFV testbed installations that usually only use a single PoP installation.

A. OSM in large Multi-PoP Environments

In the first set of experiments, we analyzed the *VIM attach* procedure, which is used to connect OSM to a single PoP using the the `osm vim-create <vim-endpoint>` command. Fig. 5 shows the total setup time breakdown to start the emulated infrastructure and to attach all emulated VIMs to OSM. The numbers behind the topologies indicate the number of nodes and links in the topology. The results show that the time required to attach the VIMs to OSM uses most of the test environment's setup time, but the system can still be deployed and configured in less than 150 s, even if the largest topology with more than 150 PoPs is used. The figure also shows the request times for all `osm vim-create` requests. It indicates that the attachment procedure becomes slightly slower when larger topologies are used.

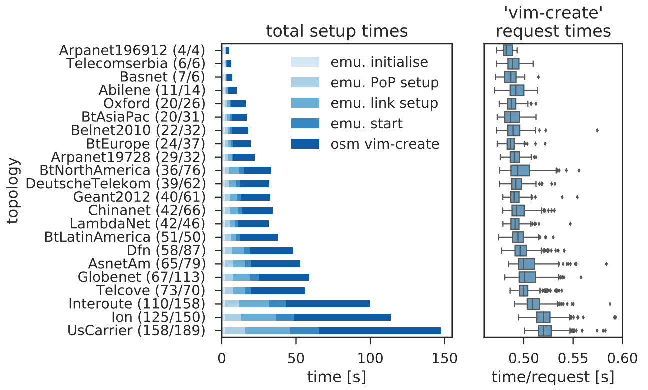


Fig. 5: Emulation setup and OSM attachment times

B. OSM Service Instantiation and Termination

In the second set of experiments, we investigated OSM's network service instantiation (`osm ns-create`) and network service termination (`osm ns-delete`) operations. To do so, we used a test network service consisting of two linked VNFs. We requested OSM to successively create 64 instances of this service, which corresponds to 128 deployed VNFs, and terminate them later, one after each other. In each instantiation request, the service was randomly placed on the available PoPs of the used topologies (Fig. 6). The results show that a service instantiation takes between 10 s and 22 s and a service termination between 5 s and 10 s. The median of the instantiation times appears to be not affected by the size of selected topology. The second plot of Fig. 6 indicates that the request times tend to slow down when more service instances are already running.

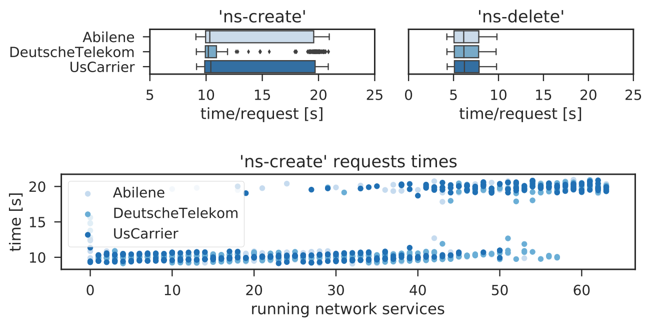


Fig. 6: OSM network service create and delete times

C. Insights and Lessons Learned

During our case study, we found and reported some interesting issues, for example, a bug that prevents a user to instantiate a network service on the 101st or higher PoPs. The reason for this is a hard-coded query limit that causes the OSM client to only fetch the first 100 PoPs that are attached to the system. This results in a *PoP not found* exception when a network service should be instantiated on, e.g., PoP 101. We also noticed that for every `osm vim-show <pop x>` command the entire VIM list is fetched by the OSM client, instead of only fetching the information of the requested PoP. This can result in increasing request delays when OSM is used with large numbers of attached PoPs.

It is important to note that such issues would not be discovered by today's NFV test deployments which usually do not use more than a handful of PoPs. But the 5G and NFV community envisions very large multi-PoP scenarios for future use cases, like IoT. As a result, MANO systems need to be tested against such large multi-PoP networks. To do this, our platform provides a flexible and easy to apply test solution that allows to verify and improve the quality of MANO systems for use cases of future networks.

V. Conclusion

Using emulation-based smoke testing as part of the automated test and integration pipeline, used by MANO software projects, contributes to the quality and production readiness of these complex software systems. The presented approach enables the pre-validation of future-readiness of MANO systems for upcoming, large-scale 5G scenarios with hundreds or thousands of PoPs. This is not possible with today's lab-scale NFV testbed installations.

The presented platform was developed as part of the European H2020 project SONATA [1] and was recently adopted by the ETSI OSM project [3], where it is maintained under the name *vim-emu* as part of the *DevOps module development group*. An outstanding example for the sustainability of European H2020 project results. It is open source and publicly available[2][3] under Apache 2.0 license.

Acknowledgments

This work has been partially supported by the SONATA project, funded by the European Commission under Grant number 671517 through the Horizon 2020 and 5G-PPP programs (www.sonata-nfv.eu), the 5GTANGO project, funded by the European Commission under Grant number H2020-ICT-2016-2 761493 through the Horizon 2020 and 5G-PPP programs (http://5gtango.eu), and the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

References

[1] SONATA Cosortium, "SONATA Project," http://sonata-nfv.eu.
[2] Linux Foundation, "ONAP: Open Network Automation Platform," https://www.onap.org.
[3] ETSI OSM, "Open Sorce MANO," https://osm.etsi.org.
[4] OpenStack Project, "DevStack," https://docs.openstack.org/devstack/latest/.
[5] E. Dustin, J. Rashka, and J. Paul, *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional, 1999.
[6] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2016 IEEE Conference on*. IEEE, 2016.
[7] A. F. Cattoni, G. C. Madueño, M. Dieudonne, P. Merino, A. D. Zayas, A. Salmeron, F. Carlier, B. Saint Germain, D. Morris, R. Figueiredo *et al.*, "An end-to-end testing ecosystem for 5g," in *Networks and Communications (EuCNC), 2016 European Conference on*. IEEE, 2016, pp. 307–312.
[8] M. Zhao, F. L. Gall, P. Cousin, R. Vilalta, R. Muoz, S. Castro, M. Peuster, S. Schneider, M. Siapera, E. Kapassa, D. Kyriazis, P. Hasselmeyer, G. Xilouris, C. Tranoris, S. Denazis, and J. Martrat, "Verification and validation framework for 5g network services and apps," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 321–326.
[9] M. Keller, C. Robbert, and M. Peuster, "An evaluation testbed for adaptive, topology-aware deployment of elastic applications," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. ACM, 2013, pp. 469–470.
[10] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-coverage Services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul 2003. [Online]. Available: http://doi.acm.org/10.1145/956993.956995
[11] H2020 SoftFIRE consortium, "SoftFIRE Approach to Experiment Management: Why and How," online at https://goo.gl/LxLfLz, 2017.
[12] H2020 Fed4Fire+ consortium, "Fed4Fire: The largest federation of testbeds in europe," https://www.fed4fire.eu.
[13] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010.
[14] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, San Diego, CA, USA, Nov 2008, pp. 1–7.
[15] L. Mamatas, S. Clayman, and A. Galis, "A service-aware virtualized software-defined infrastructure," *Communications Magazine, IEEE*, vol. 53, no. 4, pp. 166–174, 2015.
[16] M. Peuster, S. Draxler, H. R. Kouchaksaraei, S. v. Rossem, W. Tavernier, and H. Karl, "A flexible multi-pop infrastructure emulator for carrier-grade mano systems," in *Network Softwarization (NetSoft), 2017 IEEE Conference on*. IEEE, 2017, pp. 1–3.
[17] M. Peuster, "Containernet," https://containernet.github.io, 2017.
[18] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[2]ETSI OSM *vim-emu* https://osm.etsi.org/gitweb/?p=osm/vim-emu.git;
[3]ETSI OSM *vim-emu* documentation: https://goo.gl/XZNLVw