Joint testing and profiling of microservice-based network services using TTCN-3

Manuel Peuster*, Christian Dröge, Clemens Boos, Holger Karl

Computer Networks Group, Paderborn University, Warburgerstr. 100, 33098 Paderborn, Germany

Abstract

The ongoing softwarization of networks creates a big need for automated testing solutions to ensure service quality. This becomes even more important if agile environments with short time to market and high demands, in terms of service performance and availability, are considered. In this paper, we introduce a novel testing solution for virtualized, microservice-based network functions and services, which we base on TTCN-3, a well known testing language defined by the European standards institute (ETSI). We use TTCN-3 not only for functional testing but also answer the question whether TTCN-3 can be used for performance profiling tasks as well. Finally, we demonstrate the proposed concepts and solutions in a case study using our open-source prototype to test and profile a chained network service.

Keywords: network function virtualization, service chains, automated testing, performance profiling, test definition

1. Introduction

Automated testing of microservice-based virtualized network functions (VNF) as well as more complex service function chains (SFC) is still a challenge that needs to be addressed to finally deliver the full agility promised by network function virtualization (NFV) and network softwarization [1]. Those tests are not only limited to functional, integration, or compatibility tests but must also include a variety of performance tests to learn about the behavior of VNFs and SFCs under different resource configurations — a methodology called *NFV profiling* [2, 3, 4].

Most NFV profiling solutions consider the profiled VNFs or SFCs as black-boxes that can be configured from the outside, e.g., by assigning resources like CPU cores or memory to them. Profiling then happens by deploying the VNF or SFC with a certain configuration and stimulating it, e.g., by sending generated traffic to it, while its resulting performance, e.g., throughput or latency, is measured. This process is then repeated for all possible configurations resulting in a parameter study producing a so-called *VNF or SFC performance profile* as output [4].

In this paper we approach the automated VNF and SFC testing and profiling challenge using a testing solution well known in the ICT industry: The *Testing and Test Control Notation Version 3 (TTCN-3)* [5]. TTCN3 is standardized by ETSI and is a modular testing language with special focus on reusable, automated and distributed testing of communication systems. It is the de-facto standard for compliance testing in the ICT field and used by standardization bodies like 3GPP and IETF. We believe that TTCN-3 is well suited for use in the NFV landscape, as we show in Section 3 where we present a solution to integrate TTCN-3 with the ETSI-defined NFV architecture. This creates an automated NFV testing solution, which is our *first contribution*. Our *second contribution* relies on the idea that once we can use TTCN-3 for NFV testing, we should make use of it for NFV profiling as well. We argue that this is a natural fit, since test developers — already familiar with TTCN-3 — are often responsible to define SFC profiling experiments. To this end, we introduce the *TTCN-3 service profiling format (TSPF)* in Section 4 before evaluating the applicability of the presented concepts with a case study in Section 5.

2. Related Work

Standardization bodies, like ETSI, specify guidelines for testing an NFV infrastructure focusing mainly on performance and compatibility aspects [6]. Some IETF work, in contrast, focusses on the verification of SFC forwarding graphs [7], e.g., loop detection.

*Corresponding author:
Email address: manuel.peuster@uni-paderborn.de (Holger Karl)

None of them focusses on practical solutions for VNF and SFC testing as we do in this paper. Another approach for NFV testing is using testing platforms, like 5GTANGO's verification and validation (V&V) platform [8] or ONAP's VNF test platform (VTP) [9], which focus more on VNF and SFC certification aspects but can be combined with the solution proposed in this paper. Even if the existing NFV profiling solutions [2, 3] already offer a high degree of automation, their experiments are still defined using custom-tailored experiment descriptions rather than a common description approach. None of these solutions support the joint definition of tests and profiling experiments.

3. Combining TTCN-3 and NFV

The main idea of our work is to exploit the flexibility and abstract nature of TTCN-3 to build a generic test framework for NFV deployments and, more specifically, for single VNFs as well as complex SFCs. Those tests can be simple functional tests, using TTCN-3's flexible port and codec mechanisms to generate test packets to be sent through the SFC and observe the results, e.g., whether the correct packets are blocked or whether packet headers have been modified correctly. These tests can also utilize so-called *test agents* to generate test traffic and stimulate the SFC under test (SFC-UT), e.g., replay traffic traces. During a test, a large set of system metrics, including infrastructure metrics and metrics of the SFC-UT, are monitored and used as test results.

In classical test setups, the test system is deployed and running before the TTCN-3 test system connects to it and executes the tests. This approach could directly be applied to the presented NFV use cases by manually instructing a management and orchestration (MANO) system to deploy the SFC-UT and then executing the tests against it. However, such an approach would leave one of the main features of the NFV concept unexploited — *automated deployments*. To this end, we designed the proposed approach such that the TTCN-3 test system itself can trigger the deployment and initial configuration of the SFC-UT before executing the tests against it, allowing end-to-end automation of test procedures. To do so, we integrate the TTCN-3 test system with NFV deployments using an abstraction layer consisting of multiple adaptation components. Figure 1 presents a schematic overview of the proposed system architecture and highlights this integration.

On top, the *test system user* is shown who interacts with the *TTCN-3 test system* to define, implement, or trigger tests. We use a simplified TTCN-3 test system

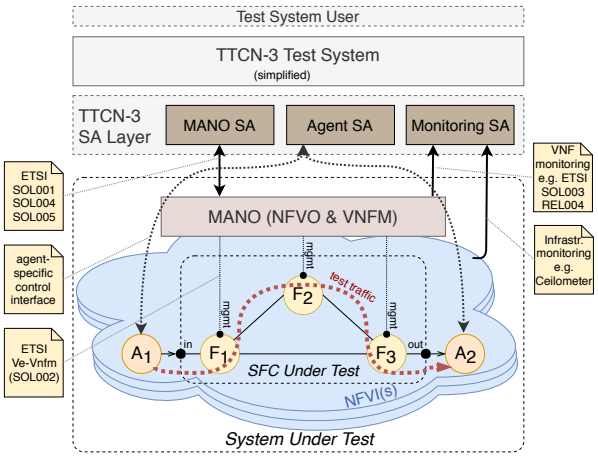

Figure 1: System architecture applying TTCN-3 to an ETSI-compatible NFV system using a set of re-usable and standard compliant system adaptors.

box in this figure and hide internal components (see TTCN-3 literature [5]), because they are not relevant for the proposed architecture. Below the test system, the *TTCN-3 system adapter layer* is shown, containing three system adaptors (SA). These SAs are one part of our contribution and are used to connect the TTCN-3 test system with the NFV infrastructure (NFVI), MANO system, and to interface with the VNFs or SFC under test. This builds an abstraction layer and allows test cases, implemented in TTCN-3 and executed by the test system, to be designed independently of a particular NFV target paltform.

The first SA is the *MANO SA* that interfaces with the northbound interfaces of an MANO system and controls the lifecycle, i.e., on-boarding, instantiation, configuration, and termination of the VNFs and SFCs under test. More specifically, it implements multiple ETSI SOL interfaces [10] so that it can be used with different ETSI-compliant MANO solutions. SAs that implement custom (or proprietary) MANO interfaces are still possible. The *Agent SA* is the second adaptor in the proposed system and is responsible to control *test agents*, e.g., A_1 and A_2, that are deployed together with the SFC-UT. Those agents are active components used to stimulate the tested systems, e.g., by generating traffic, like shown in the figure. Besides triggering those active stimuli, agents do also report results, e.g., response time measurements, which are sent back to TTCN-3. It is possible to have multiple *Agent SAs* in a system to interface with different kinds of agents. Finally, the third adaptor, called *Monitoring SA*, is used for passive monitoring of the system. It utilizes multiple sources to collect monitoring data. First, monitoring data is collected from

the NFVI, e.g., system resource usage or traffic counters which can, e.g., be used to detect packet loss in an SFC. Second, monitoring data is forwarded by the MANO system, e.g., application level metrics of the deployed VNFs or SFCs. It is worth noting that the collection of additional monitoring data is entirely optional and can be considered a helper for the realization of more sophisticated test cases. At the bottom of Figure 1, the actual NFV deployment with one or multiple connected NFVIs is shown on which the tested VNFs or SFCs are deployed and executed.

The presented system can either be used directly, e.g., in a local test environment of a developer or it can be integrated into an NFV testing platform, like the 5GTANGO V&V [8] or ONAP VTP [9].

4. TTCN-3-based function & service profiling

Having the presented TTCN-3-based testing solution for NFV in place, our next goal was to use it as automation solution to perform VNF and SFC profiling experiments, i.e., defining a set of performance tests with different configurations and automatically execute them [2]. To achieve this, we designed and specified the *TSPF*, which is the second major contribution of this paper. Listing 1 shows a shortened example of a TSPF profiling experiment definition as we use it in our case study presented in Section 5.

```
1  module TSPF_wrk_haproxy_nginx_full_profile {
2    // profiling experiment definition
3    template TSP_Experiment my_example := {
4      name := "wrk-haproxy-nginx",
5      repetitions := 10,
6      service_name := "de.upb.haprx-ngx-service"};
7    // parameters to be tested
8    template ParameterConf paramcfg := {{
9      function_id := "haproxy-vnf",
10     vcpus := { 1,2,3,4 },
11     cpu_time := { 0.1, 0.5, 1.0 },
12     memory := { 128, 256 }
13   }}; // ... definition for further VNFs of SFC
14   // agent setup (reusing wrk_client template)
15   template Agents agent_list := {
16     wrk_client("haproxy-vnf:input", "dc1",
17       "wrk http://${VNF:IP4:haproxy-vnf}")};
18 } // ... control, monitoring conf. (skipped)
```

Listing 1: Example TSPF profiling definition using TTCN-3

We designed TSPF using the following requirements and design principles. *(i) pure:* To make it as reusable as possible, TSPF is completely based on TTCN-3 and does not require any language extensions. *(ii) complete:* It allows to define and configure all relevant aspects of a profiling experiment and the test system shown in Figure 1, including agents, monitors, and the configurations of the SFC-UT. *(iii) modular:* Each part of a TSPF experiment can be re-used in other experiments, e.g., agent descriptions and configurations can be shared by multiple experiments. *(iv) compact:* Configuration values can be specified as lists or iterators (line 10–12) to ease the definition of complex parameter studies to be executed. *(v) deployment agnostic:* Placeholder macros allow to define experiments independently of the actual deployment of the profiled SFC and its VNFs (line 17).

Each TSPF profiling experiment definition consists of five sections (realized as reusable TTCN-3 templates). First, the experiment header (line 3), defining global properties, like number of repetitions. Second, the parameters to be tested during the experiment for each involved VNF (line 8). Third, the definition of the used agents and their properties (line 15). Fourth, the configuration of the involved monitoring systems and finally, the control section used as entry point for TTCN-3 (both skipped in the example). The example in Listing 1 demonstrates how compact the definition of a complex profiling experiment can be and that it is possible to capture all relevant information to automate profiling experiments using TTCN-3.

5. Case Study

We performed a case study to verify that our proposed TTCN-3-based NFV profiling approach works and to get insights about its performance and scalability. To do so, we use an example SFC which consists of a *HAProxy* VNF and an *Nginx* VNF. We use the traffic generator *Wrk* as a test agent to stimulate the SFC. All these components are implemented as Docker containers and we use *vim-emu* [11] as test backend and MANO solution. The SFC is profiled using the configuration parameters shown in Listing 1, namely 1 to 4 CPU cores, 10%, 50%, and 100% assigned CPU time, as well as 128 MiB and 256 MiB memory, resulting in 24 tested configurations per VNF and thus 576 tested configurations for the complete SFC. Each configuration was tested 10 times resulting in 5760 overall profiling rounds in each performed experiment. All scenarios were executed on host systems with Intel Core i7 960 CPU @ 3.20 GHz and 24 GiB RAM running Ubuntu 16.04.3 LTS with kernel 4.4 and Titan [5] toolset version 6.4.0.

Figure 2 reports a subset of the overall profiling results, showing (a) the number of requests performed per second as well as (b) the latency 99th percentile of requests sent to the Nginx VNF through the HAProxy VNF. The results are averaged over 10 s experiment time and 10 repetitions. They are shown in relation to the CPU times assigned to the two involved VNFs, each

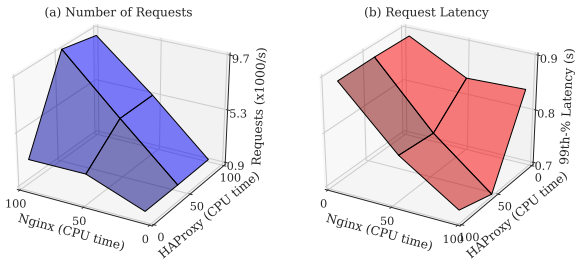

Figure 2: SFC performance over assigned CPU time

Conf. count	Runtime	Max. Mem.	MANO SA
10	378.873 s	21208 KiB	vim-emu
100	3801.941 s	21040 KiB	vim-emu
1000	38773.167 s	20972 KiB	vim-emu

Table 1: Runtime and maximum memory usage

with 1 CPU and 128 MiB memory. Results for the remaining SFC configurations as well as further request sizes are available online [12].

It can be seen that the CPU time assigned to the Nginx VNF has the most impact on the overall SFC performance. The CPU time assigned to HAProxy has only an impact if it is small and the CPU time available for Nginx is high. This performance behavior information represents a so called *SFC performance profile* and can be used to optimize resource dimensioning decisions in production deployments [2]. The important point of this study are not the absolute performance values achieved by the used SFC but the fact that those performance figures are derived by a *fully automated profiling process* without any further human interaction. This shows a clear use case of the presented profiling approach, allowing to use TTCN-3 not only for functional testing of VNFs and SFCs, but also for performance tests and to automatically construct complex performance profiles.

Table 1 quantifies the resource demands (in terms of runtime and memory consumption) of our prototype. The table indicates that the runtime shows a linear relationship to the number of tested configurations (profiling rounds). Still, the presented architecture allows to speed up experiments by using multiple NFVIs in parallel. The table shows that memory usage does not depend on the number of tested configurations, which is important for the overall scalability of the system.

6. Conclusion

The presented testing concepts drastically reduce the complexity of using TTCN-3 as NFV test solution. We showed that TTCN-3 can also be used for performance profiling tasks, although the syntax of the resulting experiment definitions can still be simplified, which we leave for future work. The presented case study shows the applicability of the presented concepts and how they can be used to automate performance profiling tasks. Our prototype and raw measurements are available online [12].

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 27 (9) (2016) 1206–1215.

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

[3] R. V. Rosa, C. Bertoldo, C. E. Rothenberg, Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking, IEEE Communications Magazine 55 (9) (2017) 110–117. doi:10.1109/MCOM.2017.1700127.

[4] R. V. Rosa, C. E. Rothenberg, M. Peuster, H. Karl, Methodology for VNF Benchmarking Automation, Internet-Draft draft-rosa-bmwg-vnfbench-02, Internet Engineering Task Force (IETF), work in Progress (July).

[5] ETSI, 201 873-1 (V3.1.1): Methods for Testing and Specification (MTS): The Testing and Test Control Notation version 3 (2005).

[6] ETSI GS NFV-TST 001, Network Functions Virtualization (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services (2016).

[7] M.-K. Shin, K.-H. Nam, S. Pack, S. Lee, R. Krishnan, Verification of NFV Services: Problem Statement and Challenges, Internet-Draft draft-irtf-nfvrg-service-verification-05, Internet Engineering Task Force (IETF), work in Progress (October).

[8] P. Twamley, M. Müller, P.-B. Bök, G. K. Xilouris, C. Sakkas, M. A. Kourtis, M. Peuster, S. Schneider, P. Stavrianos, D. Kyriazis, 5GTANGO: An Approach for Testing NFV Deployments, in: 2018 European Conference on Networks and Communications (EuCNC), IEEE, 2018, pp. 1–218.

[9] Linux Foundation, ONAP: VNF Test Platform, online at: `https://bit.ly/2UPcwEd` (2018).

[10] ETSI, Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification, Website, online at `https://www.etsi.org` (2018).

[11] M. Peuster, H. Karl, 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. doi:10.1109/NFV-SDN.2016.7919490.

[12] C. Droege, TTCN-3 Service Profiling, online at: `https://git.io/fpAWY` (2018).