BMWG
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

R. Rosa, Ed.
C. Rothenberg
UNICAMP
M. Peuster
H. Karl
UPB

July 2, 2018

Methodology for VNF Benchmarking Automation
draft-rosa-bmwg-vnfbench-02

Abstract

This document describes a common methodology for automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of benchmarking methodologies for particular VNFs can be derived from this document. Two open source reference implementations are reported as running code embodiments of the proposed, automated benchmarking methodology.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Benchmarking Methodology Working Group (BMWG) initiated efforts, approaching considerations in [RFC8172], to develop methodologies for benchmarking VNFs. Similarly described in [RFC8172], VNF benchmark motivating aspects define: (i) pre-deployment infrastructure dimensioning to realize associated VNF performance profiles; (ii) comparison factor with physical network functions; (iii) and output results for analytical VNF development.

Having no strict and clear execution boundaries, different from earlier self-contained black-box benchmarking methodologies described in BMWG, a VNF depends on underlying virtualized environment parameters [ETS14a], intrinsic factors to be analyzed when one investivages the performance of a VNF. This document stands as a ground methodology guide for VNF benchmarking automation. It addresses the state-of-the-art publications and the current developments in similar standardization efforts (e.g., [ETS14c] and [RFC8204]) towards bechmarking VNFs.

Automating the extraction of VNF performance metrics propitiates: (i) the development of agile performance-focused DevOps methodologies for Continuous Integration and Delivery (CI/CD) of VNFs; (ii) the creation of on-demand VNF test descriptors for upcoming execution environments; (iii) the path for precise-analytics of extensively automated catalogues of VNF performance profiles; (iv) and run-time profiling mechanisms to assist VNF lifecycle orchestration/management workflows.

2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. Also, the reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETS14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - The principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

NFVI PoP: NFV Infrastructure Point of Presence - Any combination of virtualized compute, storage and network resources.

NFVI: NFV Infrastructure - Collection of NFVI PoPs under one orchestrator.

VIM: Virtualized Infrastructure Manager - functional block that is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator’s Infrastructure Domain (e.g. NFVI-PoP).

VNFM: Virtualized Network Function Manager - functional block that is responsible for controlling and managing the VNF life-cycle.

NFVO: NFV Orchestrator - functional block that manages the Network Service (NS) life-cycle and coordinates the management of NS life-cycle, VNF life-cycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.

VNF: Virtualized Network Function - a software-based network function. A VNF can be either represented by a single entity or be composed by a set of smaller, interconnected software components, called VNF components (VNFCs) [ETS14d]. Those VNFs are also called composed VNFs.

VNFD: Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF on-boarding and managing the life cycle of a VNF instance.

VNFC: Virtualized Network Function Component - a software component that implements (parts of) the VNF functionality. A VNF can consist of a single VNFC or multiple, interconnected VNFCs [ETS14d]

VNF-FG: Virtualized Network Function Forwarding Graph - an ordered list of VNFs or VNFCs creating a service chain.

3. Scope

This document assumes VNFs as black boxes when defining their benchmarking methodologies. White box approaches are assumed and analysed as a particular case under the proper considerations of internal VNF instrumentation, later discussed in this document.

In what follows, this document outlines a basis methodology for VNF benchmarking, specifically addressing its automation.

4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETS14c].

4.1. VNF Testing Methods

Following the ETSI’s model in [ETS14c], we distinguish three methods for VNF evaluation:

Benchmarking: Where parameters (e.g., cpu, memory, storage) are provided and the corresponding performance metrics (e.g., latency, throughput) are obtained. Note, such request might create multiple reports, for example, with minimal latency or maximum throughput results.

Verification: Both parameters and performance metrics are provided and a stimulus verify if the given association is correct or not.

Dimensioning: Where performance metrics are provided and the corresponding parameters obtained. Note, multiple deployment interactions may be required, or if possible, underlying allocated resources need to be dynamically altered.

Note: Verification and Dimensioning can be reduced to Benchmarking. Therefore, we detail Benchmarking in what follows.

4.2. Generic VNF Benchmarking Setup

A generic VNF benchmarking setup is shown in Figure 1, and its components are explained below. Note here, not all components are mandatory, and VNF benchmarking scenarios, further explained, can dispose its components in varied settings.

```
                              +---------------+
                              |    Manager    |
                Control       | (Coordinator) |
                Interface     +---+-------+---+
             +--------+-----------+       +-------------------+
             |        |                                       |
             |        |   +-------------------------+         |
             |        |   |    System Under Test    |         |
             |        |   |                         |         |
             |        |   |    +-----------------+  |         |
             |     +--+------- +       VNF       |  |         |
             |     |           |                 |  |         |
             |     |           | +----+   +----+ |  |         |
             |     |           | |VNFC|...|VNFC| |  |         |
             |     |           | +----+   +----+ |  |         |
             |     |           +----.---------.--+  |         |
       +-----+---+ |  Monitor  |    :         :     |   +-----+----+
       | Agent   | |{listeners}|----^---------V--+  |   |  Agent   |
       |(Sender) | |           |    Execution    |  |   |(Receiver)|
       |         | |           |   Environment   |  |   |          |
       |{Probers}| +-----------|                 |  |   |{Probers} |
       +-----.---+        |    +----.---------.--+  |   +-----.----+
             :            +---------^---------V-----+         :
             V                      :         :               :
             :................>.....:         :............>..:
             Stimulus Traffic Flow
```

Figure 1: Generic VNF Benchmarking Setup

Agent -- executes active stimulus using probers, i.e., benchmarking tools, to benchmark and collect network and system performance metrics. While a single Agent is capable of performing localized benchmarks in execution environments (e.g., stress tests on CPU, memory, disk I/O), the interaction among distributed Agents enable the generation and collection of VNF end-to-end metrics (e.g., frame loss rate, latency). In a benchmarking setup, one Agent can create the stimuli and the other end be the VNF itself where, for example, one-way latency is evaluated. An Agent can be defined by a physical or virtual network function.

Prober -- defines a software/hardware-based tool able to generate stimulut traffic specific to a VNF (e.g., sipp) or generic to multiple VNFs (e.g., pktgen). A Prober must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of stilumi, parsing of extracted metrics, and debugging options. Specific Probers might be developed to abstract and to realize the description of particular VNF benchmarking methodologies.

Monitor -- when possible, it is instantiated inside the target VNF or NFVI PoP (e.g., as a plug-in process in a virtualized environment) to perform passive monitoring, using listeners, for metrics collection based on benchmark tests evaluated according to Agents' stimuli. Different from the active approach of Agents that can be seen as generic benchmarking VNFs, Monitors observe particular properties according to NFVI PoPs and VNFs capabilities. A Monitor can be defined as a virtual network function.

Listener -- defines one or more software interfaces for the extraction of particular metrics monitored in a target VNF and/or execution environment. A Listener must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of monitoring captures, parsing of extracted metrics, and debugging options. White-box benchmarking approaches must be carefully analyzed, as varied methods of performance monitoring might be coded as a Listener, possibly impacting the VNF and/or execution environment performance results.

Manager -- in a VNF benchmarking setup, a Manager is responsible for (i) the coordination and synchronization of activities of Agents and Monitors, (ii) collecting and parsing all VNF benchmarking results, and (iii) aggregating the inputs and parsed benchmark outputs to construct a VNF performance profile, which defines a report that correlates the VNF stimuli and the monitored metrics. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking results. A Manager can be defined as a physical or virtual network function, and be split into multiple sub-components, each responsible for separated functional aspects of the overall Manager component.

Virtualized Network Function (VNF) -- consists of one or more software components, so called VNF components (VNFC), adequate for performing a network function according to allocated virtual resources and satisfied requirements in an execution environment. A VNF can demand particular configurations for benchmarking specifications, demonstrating variable performance profiles based on available virtual resources/parameters and configured enhancements targeting specific technologies (e.g., NUMA, SR-IOV, CPU-Pinning).

Execution Environment -- defines a virtualized and controlled composition of capabilities necessary for the execution of a VNF. An execution environment stands as a general purpose level of virtualization with abstracted resources available for one or more VNFs. It can also define specific technology habilitation, incurring in viable settings for enhancing the performance of VNFs.

4.3. Deployment Scenarios

A VNF benchmark deployment scenario establishes the physical and/or virtual instantiation of components defined in a VNF benchmarking setup.

The following considerations hold for deployment scenarios:

- Components can be composed in a single entity and be defined as black or white boxes. For instance, Manager and Agents could jointly define one hardware/software entity to perform a VNF benchmark and present results.

- Monitor is not a mandatory component and must be considered only when performed white box benchmarking approaches for a VNF and/or its execution environment.

- Monitor can be defined by multiple instances of software components, each addressing a VNF or execution environment and their respective open interfaces for the extraction of metrics.

- Agents can be disposed in varied topology setups, included the possibility of multiple input and output ports of a VNF being directly connected each in one Agent.

- All benchmarking components defined in a deployment scenario must perform the synchronization of clocks.

4.4. Influencing Aspects

In general, VNF benchmarks must capture relevant causes of performance variability. Concerning a deployment scenario, influencing aspects on the performance of a VNF can be observed in:

Deployment Scenario Topology: The disposition of components can define particular interconnections among them composing a specific case/method of VNF benchmarking.

Execution Environment: The availability of generic and specific capabilities satisfying VNF requirements define a skeleton of opportunities for the allocation of VNF resources. In addition, particular cases can define multiple VNFs interacting in the same execution environment of a benchmarking setup.

VNF: A detailed description of functionalities performed by a VNF sets possible traffic forwarding and processing operations it can perform on packets, added to its running requirements and specific configurations, which might affect and compose a benchmarking setup.

Agent: The toolset available for the benchmarking stimulus of a VNF and its characteristics of packets format and workload can interfere in a benchmarking setup. VNFs can support specific traffic format as stimulus.

Monitor: In a particular benchmarking setup where measurements of VNF and/or execution environment metrics are available for extraction, an important analysis consist in verifying if the Monitor components can impact performance metrics of the VNF and the underlying execution environment.

Manager: The overall composition of VNF benchmarking procedures can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics.

The listed influencing aspects must be carefully analyzed while automating a VNF benchmarking methodology.

5. Methodology

Portability is an intrinsic characteristic of VNFs and allows them to be deployed in multiple environments. This enables various benchmarking procedures in varied deployment scenarios. A VNF benchmarking methodology must be described in a clear and objective manner in order to allow effective repeatability and comparability of the test results. Those results, the outcome of a VNF benchmarking process, are captured in a VNF Benchmarking Report (VNF-BR) as shown in Figure 2.

```
                                             X
                                            / \
                                           /   \
                                          /     \
               +--------+                /       \
               |        |               /         \
               | VNF-BD |--(defines)-->| Benchmark |
               |        |               \ Process /
               +--------+                \       /
                                          \     /
                                           \   /
                                            \ /
                                             V
                                             |
                                        (generates)
                                             |
                                             v
                                +-------------------------+
                                |         VNF-BR          |
                                | +--------+   +--------+ |
                                | |        |   |        | |
                                | | VNF-BD |   | VNF-PP | |
                                | | {copy} |   |        | |
                                | +--------+   +--------+ |
                                +-------------------------+
```

Figure 2: VNF benchmarking process inputs and outputs

VNF Benchmarking reports comprise two parts:

VNF Benchmarking Descriptor (VNF-BD) -- contains all required definitions and requirements to configure, execute and reproduce VNF benchmarking experiments. VNF-BDs are defined by the developer of a benchmarking experiment and serve as input to the benchmarking process, before being included in the generated VNF-BR.

VNF Performance Profile (VNF-PP) -- contains all measured metrics resulting from the execution of a benchmark. Additionally, it might also contain additional recordings of configuration parameters used during the execution of the benchmarking scenario to facilitate comparability of VNF-BRs.

A VNF-BR correlates structural and functional parameters of VNF-BD with extracted VNF benchmarking metrics of the obtained VNF-PP. The content of each part of a VNF-BR is described in the following sections.

5.1. VNF Benchmarking Descriptor (VNF-BD)

VNF Benchmarking Descriptor (VNF-BD) -- an artifact that specifies a method of how to measure a VNF Performance Profile. The specification includes structural and functional instructions and variable parameters at different abstraction levels (e.g., topology of the deployment scenario, benchmarking target metrics, parameters of benchmarking components). VNF-BD may be specific to a VNF or applicable to several VNF types. A VNF-BD can be used to elaborate a VNF benchmark deployment scenario aiming at the extraction of particular VNF performance metrics.

The following items define the VNF-BD contents.

5.1.1. Procedures Configuration

The definition of parameters concerning the execution of the benchmarking procedures (see Section 5.3), for instance, containing the number of repetitions and duration of each test.

5.1.2. Target Information

General information addressing the target VNF, with references to any of its specific characteristics (e.g., type, model, version/release, architectural components, etc). In addition, it defines the metrics to be extracted when running the benchmarking tests.

5.1.3. Deployment Scenario

This section of a VNF-BD contains all information needed to describe the deployment of all involved components used during the benchmarking test.

5.1.3.1. Topology

Information about the experiment topology, concerning the disposition of the components in a benchmarking setup (see Section 4.2). It must define the role of each component and how they are interconnected (i.e., interface, link and network characteristics).

5.1.3.2. Requirements

Involves the definition of execution environment requirements to execute the tests. Therefore, they concern all required capabilities needed for the execution of the target VNF and the other components composing the benchmarking setup. Examples of specifications involve: min/max allocation of resources, specific enabling technologies (e.g., DPDK, SR-IOV, PCIE).

5.1.3.3. Parameters

Involves any specific configuration of benchmarking components in a setup described the the deployment scenario topology.

VNF Configurations: Defines any specific configuration that must be loaded into the VNF to execute the benchmarking experiments (e.g., routing table, firewall rules, vIMS subscribers profile).

VNF Resources: Contains particular VNF resource configurations that should be tested during the benchmarking process, e.g., test the VNF for configurations with 2, 4, and 8 vCPUs associated.

Agents: Defines the configured toolset of available probers and related benchmarking/active metrics, available workloads, traffic formats/traces, and configurations to enable hardware capabilities (if existent).

Monitors: defines the configured toolset of available listeners and related monitoring/passive metrics, configuration of the interfaces with the monitoring target (VNF and/or execution environment), and configurations to enable specific hardware capabilities (if existent).

5.2. VNF Performance Profile (VNF-PP)

VNF Performance Profile (VNF-PP) -- defines a mapping between resources allocated to a VNF (e.g., CPU, memory) as well as assigned configurations (e.g., routing table used by the VNF) and the VNF performance metrics (e.g., throughput, latency between in/out ports) obtained in a benchmarking test conducted using a VNF-BD. Logically, packet processing metrics are presented in a specific format addressing statistical significance (e.g., median, standard deviation, percentiles) where a correspondence among VNF parameters and the delivery of a measured VNF performance exists.

The following items define the VNF-PP contents.

5.2.1. Execution Environment

Execution environment information is has to be included in every VNF-PP and is required to describe the environment on which a benchmark was actually executed.

Ideally, any person who has a VNF-BD and its complementing VNF-PP with its execution environment information available, must be able to reproduce the same deployment scenario and VNF benchmarking tests to obtain identical VNF-PP measurement results.

If not already defined by the VNF-BD deployment scenario requirements (Section 5.1.3), for each component in the VNF benchmarking setup, the following topics must be detailed:

Hardware Specs: Contains any information associated with the underlying hardware capabilities offered and used by the component during the benchmarking tests. Examples of such specification include allocated CPU architecture, connected NIC specs, allocated memory DIMM, etc. In addition, any information concerning details of resource isolation must also be described in this part of the VNF-PP.

Software Specs: Contains any information associated with the software apparatus offered and used during the benchmarking tests. Examples include versions of operating systems, kernels, hypervisors, container image versions, etc.

Optionally, a VNF-PP execution environment might contain references to an orchestration description document (e.g., HEAT template) to clarify technological aspects of the execution environment and any specific parameters that it might contain for the VNF-PP.

5.2.2. Measurement Results

Measurement results concern the extracted metrics, output of benchmarking procedures, classified into:

VNF Processing/Active Metrics: Concerns metrics explicitly defined by or extracted from direct interactions of Agents with a VNF. Those can be defined as generic metric related to network packet processing (e.g., throughput, latency) or metrics specific to a particular VNF (e.g., vIMS confirmed transactions, DNS replies).

VNF Monitored/Passive Metrics: Concerns the Monitors’ metrics captured from a VNF execution, classified according to the virtualization level (e.g., baremetal, VM, container) and technology domain (e.g., related to CPU, memory, disk) from where they were obtained.

Depending on the configuration of the benchmarking setup and the planned use cases for the resulting VNF-PPs, measurement results can be stored as raw data, e.g., time series data about CPU utilization of the VNF during a throughput benchmark. In the case of VNFs composed of multiple VNFCs, those resulting data should be represented as vectors, capturing the behavior of each VNFC, if available from the used monitoring systems. Alternatively, more compact representation formats can be used, e.g., statistical information about a series of latency measurements, including averages and standard deviations. The exact output format to be used is defined in the complementing VNF-BD (Section 5.1).

A VNF performance profile must address the combined set of classified items in the 3x3 Matrix Coverage defined in [RFC8172].

5.3. Automated Benchmarking Procedures

VNF benchmarking offers the possibility of defining distinct aspects/ steps that may or may not be automated:

Orchestration: placement (assignment/allocation of resources) and interconnection (physical/virtual) of network function(s) and benchmark components (e.g., OpenStack/Kubernetes templates, NFV description solutions, like OSM, SONATA, ONAP) -> Defines deployment scenario.

Management/Configuration: benchmark components and VNF are configured to execute the experiment/test (e.g., populate routing table, load pcap source files in agent) -> Realizes VNF-BD settings.

Execution: Tests/experiments are executed according to configuration and orchestrated components -> Runs the VNF benchmarking cases.

Output: There might be generic VNF footprint metrics (e.g., CPU, memory) and specific VNF traffic processing metrics (e.g., transactions or throughput). Output processing must be taken into account (e.g., if sampling is applied or not) in a generic (statistics) or specific (clustering) ways -> Generates VNF-PP.

For the purposes of dissecting the execution procedures, consider the following definitions:

Trial: is a single process or iteration to obtain VNF benchmarking metrics as a singular measurement.

Test: Defines strict parameters for benchmarking components to perform one or more trials. Multiple Trials must be performed for statistical significance of the obtained benchmarking results of a Test. Each Trial must be executed following a particular deployment scenario composed by a VNF-BD. Proper measures must be taken to ensure statistic validity (e.g., independence across trials of generated load patterns).

Method: Consists of a VNF-BD, including one or more Tests to benchmark a VNF. A Method can explicitly list ranges of parameter values for the configuration of benchmarking components. Each value of such a range is to be realized in a Test. I.e., Methods can define parameter studies.

The following sequence of events composes basic, general procedures to execute a Test (as defined above).

1. A VNF-BD must be defined to be later instantiated into and executed as a deployment scenario. Such a description must contain all the structural and functional settings defined in Section 5.1. At the end of this step, the complete method of benchmarking the target VNF is defined.

2. Via an automated orchestrator or in a manual process, all the components of the VNF benchmark setup must be allocated and interconnected. VNF and the execution environment must be configured to properly address the VNF benchmark stimuli.

3. Manager, Agent(s) and Monitor(s) (if existing), must be started and configured to execute the benchmark stimuli and retrieve expected metrics captured during or at the end of each Trial. One or more trials realize the required measurements to characterize the performance behavior of a VNF according to the benchmark setup defined in the VNF-BD.

4. Output results from each obtained benchmarking test must be collected by the Manager. In an automated or manual process, intended metrics, as described in the VNF-BD, are extracted and combined to the final VNF-PP. The combination of used VNF-BD and

generated VNF-PP make up the resulting VNF benchmark report (VNF-BR).

5.4. Particular Cases

Configurations and procedures concerning particular cases of VNF benchmarks address testing methodologies proposed in [RFC8172]. In addition to the general description previously defined, some details must be taken into consideration in the following VNF benchmarking cases.

Noisy Neighbor: An Agent can assume the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the noisy workload stimulus type, frequency, virtualization level, among others, must be detailed in the VNF-BD.

Representative Capacity: An average value of workload must be specified as an Agent stimulus. Considering a long-term analysis, the VNF must be configured to properly address a desired average behavior of performance in comparison with the value of the workload stimulus.

Flexibility and Elasticity: Having the possibility of a VNF be composed by multiple components (VNFCs), internal events of the VNF might trigger changes in VNF behavior, e.g., activating functionalities associated with elasticity such as load balancing. In this sense, a detailed characterization of a VNF must be specified (e.g. the VNFs scaling state) and be contained in the VNF-PP and thus the VNF benchmarking report.

On Failures: Similar to the case before, VNF benchmarking setups must also capture the dynamics involved in VNF behavior. In case of failures, a VNF might restart itself and possibly result in an offline period (e.g., self healing). A VNF-PP and benchmarking report must capture such variation of VNF states.

White Box VNF: A benchmarking setup must define deployment scenarios to be compared with and without monitor components into the VNF and/or the execution environment, in order to analyze if the VNF performance is affected. The VNF-PP and benchmarking report must contain such analysis of performance variability, together with all the extracted VNF performance metrics.

6. Open Source Reference Implementations

There are two open source reference implementations that are build to automate benchmarking of Virtualized Network Functions (VNFs).

6.1. Gym

The software, named Gym, is a framework for automated benchmarking of Virtualized Network Functions (VNFs). It was coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa-a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFVRG and BMWG.

Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing measurements of performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas [Rosa-a], such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, tear-down).

The proposed guiding principles, elaborated in [Rosa-b], to design and build Gym can be composed in multiple practical ways for different VNF testing purposes:

- Comparability: Output of tests shall be simple to understand and process, in a human-read able format, coherent, and easily reusable (e.g., inputs for analytic applications).

- Repeatability: Test setup shall be comprehensively defined through a flexible design model that can be interpreted and executed by the testing platform repeatedly but supporting customization.

- Configurability: Open interfaces and extensible messaging models shall be available between components for flexible composition of test descriptors and platform configurations.

- Interoperability: Tests shall be ported to different environments using lightweight components.

In [Rosa-b] Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. And in [Rosa-c], a virtual switch (Open vSwitch - OVS) was the target VNF of Gym for the analysis of VNF benchmarking automation. Such articles validated Gym as a prominent open source reference implementation for VNF benchmarking tests. Such articles set important contributions as discussion of the lessons learned and the overall NFV performance testing landscape, included automation.

Gym stands as one open source reference implementation that realizes the VNF benchmarking methodologies presented in this document. Gym is being released open source at [Gym]. The code repository includes also VNF Benchmarking Descriptor (VNF-BD) examples on the vIMS and OVS targets as described in [Rosa-b] and [Rosa-c].

6.2. tng-bench

Another software that focuses on implementing a framework to benchmark VNFs is the "5GTANGO VNF/NS Benchmarking Framework" also called "tng-bench" (previously "son-profile") and was is as part of the two European Union H2020 projects SONATA NFV and 5GTANGO [tango]. Its initial ideas were presented in [Peu-a] and the system design of the end-to-end prototype was presented in [Peu-b].

Tng-bench's aims to act as a framework for the end-to-end automation of VNF benchmarking processes. Its goal is to automate the benchmarking process in such a way that VNF-PPs can be generated without further human interaction. This enables the integration of VNF benchmarking into continuous integration and continuous delivery (CI/CD) pipelines so that new VNF-PPs are generated on-the-fly for every new software version of a VNF. Those automatically generated VNF-PPs can then be bundled with the VNFs and serve as inputs for orchestration systems, fitting to the original research ideas presented in [Rosa-a] and [Peu-a].

Following the same high-level VNF testing purposes as Gym, namely: Comparability, repeatability, configurability, and interoperability, tng-bench specifically aims to explore description approaches for VNF benchmarking experiments. In [Peu-b] a prototype specification VNF-BDs is presented which not only allows to specify generic, abstract VNF benchmarking experiments, it also allows to describe sets of parameter configurations to be tested during the benchmarking process, allowing the system to automatically execute complex parameter studies on the SUT, e.g., testing a VNF's performance under different CPU, memory, or software configurations.

Tng-bench was used to perform a set of initial benchmarking experiments using different VNFs, like a Squid proxy, an Nginx load balancer, and a Socat TCP relay in [Peu-b]. Those VNFs have not only been benchmarked in isolation, but also in combined setups in which up to three VNFs were chained one after each other. These experiments were used to test tng-bench for scenarios in which composed VNFs, consisting of multiple VNF components (VNFCs), have to be benchmarked. The presented results highlight the need to benchmark composed VNFs in end-to-end scenarios rather than only benchmark each individual component in isolation, to produce meaningful VNF-PPs for the complete VNF.

Tng-bench is actively developed and released as open source tool under Apache 2.0 license [tng-bench].

7. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of VNFs in a lab environment with isolated network.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network.

Special capabilities SHOULD NOT exist in the VNF benchmarking deployment scenario specifically for benchmarking purposes. Any implications for network security arising from the VNF benchmarking deployment scenario SHOULD be identical in the lab and in production networks.

8. IANA Considerations

This document does not require any IANA actions.

9. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil. Parts of this work have received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO: https://5gtango.eu).

10. References

10.1. Normative References

[ETS14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01-_60/gs_NFV002v010201p.pdf>.

[ETS14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf>.

[ETS14c] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V1.1.1", April 2016, <http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0015.zip>.

[ETS14d] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture - ETSI GS NFV SWA001 V1.1.1", December 2014, <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf>.

[RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991, <https://www.rfc-editor.org/info/rfc1242>.

[RFC8172] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", July 2017, <https://www.rfc-editor.org/info/rfc8172>.

[RFC8204] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", September 2017, <https://www.rfc-editor.org/info/rfc8204>.

10.2. Informative References

[Gym] "Gym Home Page", <https://github.com/intrig-unicamp/gym>.

[Peu-a] M. Peuster, H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management", Fifth European Workshop on Software Defined Networks (EWSDN) , 2016, <http://ieeexplore.ieee.org/document/7956044/>.

[Peu-b] M. Peuster, H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) , 2017, <http://ieeexplore.ieee.org/document/8169826/>.

[Rosa-a] R. V. Rosa, C. E. Rothenberg, R. Szabo, "VBaaS: VNF Benchmark-as-a-Service", Fourth European Workshop on Software Defined Networks , Sept 2015, <http://ieeexplore.ieee.org/document/7313620>.

[Rosa-b] R. Rosa, C. Bertoldo, C. Rothenberg, "Take your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine Testing Series , Sept 2017, <http://ieeexplore.ieee.org/document/8030496>.

[Rosa-c] R. V. Rosa, C. E. Rothenberg, "Taking Open vSwitch to the Gym: An Automated Benchmarking Approach", IV Workshop pre-IETF/IRTF, CSBC Brazil, July 2017, <https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/rosa2017taking.pdf>.

[tango] "5GTANGO: Development and validation platform for global industry-specific network services and apps", <https://5gtango.eu>.

[tng-bench] "5GTANGO VNF/NS Benchmarking Framework", <https://github.com/sonata-nfv/tng-sdk-benchmark>.

Authors' Addresses

Raphael Vicente Rosa (editor)
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: rvrosa@dca.fee.unicamp.br
URI: https://intrig.dca.fee.unicamp.br/raphaelvrosa/

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: chesteve@dca.fee.unicamp.br
URI: http://www.dca.fee.unicamp.br/~chesteve/

Manuel Peuster
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: manuel.peuster@upb.de
URI: http://go.upb.de/peuster

Holger Karl
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: holger.karl@upb.de
URI: https://cs.uni-paderborn.de/cn/