BMWG                                                        R. Rosa, Ed.
Internet-Draft                                             C. Rothenberg
Intended status: Informational                                   UNICAMP
Expires: December 26, 2019                                    M. Peuster
                                                                 H. Karl
                                                                     UPB
                                                           June 24, 2019

# Methodology for VNF Benchmarking Automation
draft-rosa-bmwg-vnfbench-04

Abstract

This document describes a common methodology for the automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of automated 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.

Table of Contents

# 1. Introduction

The Benchmarking Methodology Working Group (BMWG) already presented considerations for benchmarking of VNFs and their infrastructure in [RFC8172]. Similar to the motivation given in [RFC8172], the following aspects justify the need for VNF benchmarking: (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.

Even if many methodologies already described by the BMWG, e.g., self-contained black-box benchmarking, can be applied to VNF benchmarking scenarios, further considerations have to be made. This is, on the one hand, because VNFs, which are software components, do not have strict and clear execution boundaries and depend on underlying virtualization environment parameters as well as management and orchestration decisions [ETS14a]. On the other hand, can and should the flexible, software-based nature of VNFs be exploited to fully automate the entire benchmarking procedure end-to-end. This is an inherent need to align VNF benchmarking with the agile methods enabled by the concept of Network Functions Virtualization (NFV) [ETS14e]. More specifically it allows: (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 automated catalogues of VNF performance profiles; (iv) and run-time mechanisms to assist VNF lifecycle orchestration/management workflows, e.g., automated resource dimensioning based on benchmarking insights.

This document describes basic methodologies and guidelines to fully automate VNF benchmarking procedures, without limiting the automated process to a specific benchmark or infrastructure. After presenting initial considerations, the document first describes a generic architectural framework to setup automated benchmarking experiments. Second, the automation methodology is discussed, with a particular focus on experiment and procedure description approaches to support reproducibility of the automated benchmarks, a key challenge in VNF benchmarking. Finally, two independent, open-source reference implementations are presented. The document addresses state-of-the-art work on VNF benchmarking from scientific publications and current developments in other standardization bodies (e.g., [ETS14c] and [RFC8204]) wherever possible.

## 2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. 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.

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.

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]

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.

NS: Network Service - a collection of interconnected VNFs forming a end-to-end service. The interconnection is often done using chaining of functions.

## 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.

This document outlines a 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 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 evaluations might create multiple reports, for example, with minimal latency or maximum throughput results.

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

Dimensioning: Performance metrics are provided and the corresponding parameters obtained. Note, multiple deployments 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 focus on Benchmarking in the rest of the document.

### 4.2. Benchmarking Procedures

A (automated) benchmarking procedure can be divided into three sub-procedures:

Trial: Is a single process or iteration to obtain VNF performance metrics from benchmarking measurements. A Test should always run multiple Trials to get statistical confidence about the obtained measurements.

Test: Defines unique structural and functional parameters (e.g., configurations, resource assignment) for benchmarked components to perform one or multiple Trials. Each Test must be executed following a particular benchmarking scenario composed by a Method. Proper measures must be taken to ensure statistical validity (e.g., independence across Trials of generated load patterns).

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

In general, automated VNF benchmarking Tests must capture relevant causes of performance variability. To dissect a VNF benchmarking

Test, in the sections that follow different benchmarking phases are categorized defining generic operations that may be automated. When automating a VNF benchmarking methodology, all the influencing aspects on the performance of a VNF must be carefully analyzed and comprehensively reported, in each phase of the overall benchmarking process.

#### 4.2.1. Phase I: Deployment

The placement (i.e., assignment and allocation of resources) and the interconnection, physical and/or virtual, of network function(s) and benchmarking components can be realized by orchestration platforms (e.g., OpenStack, Kubernetes, Open Source MANO). In automated manners, the realization of a benchmarking testbed/scenario through those means usually rely on network service templates (e.g., TOSCA, Heat, YANG). Such descriptors have to capture all relevant details of the execution environment to allow the benchmarking framework to correctly instantiate the SUT as well as helper functions required for a Test.

#### 4.2.2. Phase II: Configuration

The configuration of benchmarking components and VNFs (e.g., populate routing table, load PCAP source files in source of traffic stimulus) to execute the Test settings can be realized by programming interfaces in an automated way. In the scope of NFV, there might exist management interfaces to control a VNF during a benchmarking Test. Likewise, infrastructure or orchestration components can establish the proper configuration of an execution environment to realize all the capabilities enabling the description of the benchmarking Test. Each configuration registry, its deployment timestamp and target, must all be contained in the VNF benchmarking report.

#### 4.2.3. Phase III: Execution

In the execution of a benchmarking Test, the VNF configuration can be programmed to be changed by itself or by a VNF management platform. It means that during a Trial execution, particular behaviors of a VNF can be automatically triggered, e.g., auto-scaling of its internal components. Those must be captured in the detailed procedures of the VNF execution and its performance report. I.e., the execution of a Trial can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics. For instance, in a particular benchmarking case where the monitoring measurements of the VNF and/or execution environment are available for extraction, Tests should be run to verify if the monitoring of the VNF and/or execution environment can impact the VNF performance metrics.

#### 4.2.4. Phase IV: Report

The report of a VNF benchmarking Method might contain generic metrics (e.g., CPU and memory consumption) and VNF-specific traffic processing metrics (e.g., transactions or throughput), which can be stored and processed in generic or specific ways (e.g., by statistics or machine learning algorithms). If automated procedures are applied over the generation of a benchmarking report, those must be detailed in the report itself, jointly with their input raw measurements and output processed data. I.e., any algorithm used in the generation of processed metrics must be disclosed in the report.

## 5. Generic VNF Benchmarking Architectural Framework

A generic VNF benchmarking architectural framework, shown in Figure 1, establishes the disposal of essential components and control interfaces, explained below, that enable the automation of VNF benchmarking methodologies.

```
                              +---------------+
                              |    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

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 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.

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

Prober -- defines an abstraction layer for a software or hardware tool able to generate stimulus traffic to a VNF or perform stress tests on execution environments. Probers might be specific or generic to an execution environment or a VNF. For an Agent, a Prober must provide programmable interfaces for its life cycle management, e.g., configuration of operational parameters, execution of stilumus, 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 is instantiated inside the System Under Test, VNF and/or infrastructure (e.g., as a plug-in process in a virtualized execution environment), to perform the passive monitoring, using Listeners, for the extraction of metrics while Agents` stimuli takes place. Monitors observe particular properties according to the execution environment and VNFs capabilities, i.e., exposed passive monitoring interfaces. Multiple Listeners can be executed at once in synchrony with a Prober' stimulus on a SUT. A Monitor can be defined as a virtualized network function.

Listener -- defines one or more software interfaces for the extraction of 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 passive monitoring captures, parsing of extracted metrics, and debugging options. Varied methods of passive performance monitoring might be implemented as a Listener, depending on the interfaces exposed by the VNF and/or execution environment.

Manager -- performs (i) the discovery of available Agents/Monitors and their respective features (i.e., available Probers/Listeners and execution environment capabilities), (ii) the coordination and synchronization of activities of Agents and Monitors to perform a benchmarking Test, (iii) the collection, processing and aggregation of all VNF benchmarking measurements that correlates the VNF stimuli and the, possible, SUT monitored metrics. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking report. A Manager can be defined as a physical or virtualized network function.

### 5.1. Deployment Scenarios

A deployment scenario realizes the actual instantiation of physical and/or virtual components of a Generic VNF Benchmarking Architectural Framework needed to habilitate the execution of an automated VNF benchmarking methodology. The following considerations hold for a deployment scenario:

- Not all components are mandatory for a Test, possible to be disposed in varied settings.

- 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 benchmarking Test and present measurement results.

- Monitor can be defined by multiple instances of software components, each addressing a VNF or execution environment.

- 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.

## 6. Methodology

Portability is an intrinsic characteristic of VNFs and allows them to be deployed in multiple environments. This enables various benchmarking setups in varied deployment scenarios. A VNF benchmarking methodology must be described in a clear and objective manner following four basic principles:

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

o Repeatability: A 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.

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

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

```
                               ______________
  +--------+                  |              |
  |        |                  |  Automated   |
  | VNF-BD |--(defines)-->    | Benchmarking |
  |        |                  | Methodology  |
  +--------+                  |______________|
                                     V
                                     |
                                (generates)
                                     |
                                     v
                       +-------------------------+
                       |          VNF-BR         |
                       | +--------+   +--------+ |
                       | |        |   |        | |
                       | | VNF-BD |   | VNF-PP | |
                       | | {copy} |   |        | |
                       | +--------+   +--------+ |
                       +-------------------------+
```

Figure 2: VNF benchmarking process inputs and outputs

As shown in Figure 2, the outcome of an automated VNF benchmarking methodology, must be captured in a VNF Benchmarking Report (VNF-BR), consisting of two parts:

VNF Benchmarking Descriptor (VNF-BD) -- contains all required definitions and requirements to deploy, configure, execute, and reproduce VNF benchmarking tests. VNF-BDs are defined by the developer of a benchmarking methodology 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 benchmarking. 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.

## 6.1. 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). A VNF-BD may be specific to a VNF or applicable to several VNF types. It 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.

### 6.1.1. Descriptor Headers

The definition of parameters concerning the descriptor file, e.g., its version, identidier, name, author and description.

### 6.1.2. Target Information

General information addressing the target VNF(s) the VNF-BD is applicable, with references to any specific characteristics, i.e., the VNF type, model, version/release, author, vendor, architectural components, among any other particular features.

### 6.1.3. Experiments

The specification of the number of executions for Trials, Tests and Method. The execution of a VNF-BD corresponds to the execution of the specified Method.

### 6.1.4. Environment

The details referring to the name, description, and information associated with the interfaces needed for the orchestration, if necessary, of the specified VNF-BD scenario. I.e., it refers to a specific interface that receives the VNF-BD scenario information and converts it to the template needed for an orchestration platform. In this case, the means to the Manager component interface such orchestration platform must be provided, as well as its outcome orchestration status information (e.g., management interfaces of deployed components).

#### 6.1.5. Scenario

This section contains all information needed to describe the deployment of all involved functional components mandatory for the execution of the benchmarking Tests addressed by the VNF-BD.

##### 6.1.5.1. Nodes

Information about each component in a benchmarking setup (see Section 5). It contains the identification, name, image, role (i.e., agent, monitor, sut), connection-points and resource requirements (i.e., allocation of cpu, memory, disk).

The lifecycle specification of a node lists all the workflows that must be realized on it during a Test. For instance, main workflows include: create, start, stop, delete. Particular workflows can be specified containing the required parameters and implementation. Those details must reflect the actions taken on or by a node that might affect the VNF performance profile.

##### 6.1.5.2. Links

Links contain information about the data plane links interconnecting the components of a benchmarking setup. Links refer to two or more connection-points of a node. A link might refer to be part of a network. Depending on the link type, the network might be implemented as a layer 2 mesh, or as directional-oriented traffic forwarding flow entries. Links also detain resource requirements, specifying the minimum bandwidth, latency, and frame loss rate for the execution of benchmarking Tests.

##### 6.1.5.3. Policies

Involves the definition of execution environment policies to run the Tests. Policies might specify the (anti-)affinity placement rules for each component in the topology, min/max allocation of resources, and specific enabling technologies (e.g., DPDK, SR-IOV, PCIE) needed for each component.

### 6.1.6. Proceedings

This information is utilized by the Manager component to execute the benchmarking Tests. It consists of agent(s) and monitor(s) settings, detailing their prober(s)/listener(s) specification and running parameters.

Agents: Defines a list containing the Agent(s) needed for the VNF-BD tests. The information of each Agent contains its host environment, making reference to a node specified in the VNF-BD scenario (Section 6.1.5). In addition, each Agent also is defined with the the configured toolset of the Prober(s) and their running parameters fulfilled (e.g., stimulus workload, traffic format/trace, configurations to enable hardware capabilities, if existent). In each Prober, it is also detailed the output metrics to be extracted from it when running the benchmarking Tests.

Monitors: Defines a list containing the Monitor(s) needed for the VNF-BD tests. The information of each Monitor contains its host environment, making reference to a node specified in the VNF-BD scenario (Section 6.1.5) and detailing the placement settings of it (e.g., internal or external with the target VNF and/or execution environment). In addition, each Monitor also is defined with the the configured toolset of the Listener(s) and their running parameters fulfilled (e.g., tap interfaces, period of monitoring, interval among the measurements). In each Listener, it is also detailed the output metrics to be extracted from it when running the benchmarking Tests.

## 6.2. 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, CPU, memory) 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.

### 6.2.1. Execution Environment

Execution environment information has to be included in every VNF-PP and is required to describe the environment on which a benchmark Test 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 6.1.5), for each component in the deployment scenario of 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.

#### 6.2.2. 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., HTTP 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 6.1).

The representation format of a VNF-PP must be easily translated to address the combined set of classified items in the 3x3 Matrix Coverage defined in [RFC8172].

### 6.3. Procedures

The methodology for VNF Benchmarking Automation encompasses the process defined in Figure 2, i.e., the procedures that translate a VNF-BD into a VNF-PP composing a VNF-BR by the means of the components specified in Figure 1. This section details the sequence of events that realize such process.

#### 6.3.1. Pre-Execution

Before the execution of benchmarking Tests, some procedures must be performed:

1. A VNF-BD must be defined to be later instantiated into a deployment scenario and have executed its Tests. Such a description must contain all the structural and functional settings defined in Section 6.1. At the end of this step, the complete Method of benchmarking the target VNF is defined.

2. The VNF target image must be prepared to be benchmarked, having all its capabilities fully described. In addition all the probers and listeners defined in the VNF-BD must be implemented to realize the benchmark Tests. At the end of this step, the complete set of components of the benchmarking VNF-BD deployment scenario is defined.

3. The environment needed for a VNF-BD must be defined to realize its deployment scenario, in an automated or manual method. This step might count on the instantiation of orchestration platforms and the composition of specific topology descriptors needed by those platforms to realize the VNF-BD deployment scenario. At the end of this step, the whole environment needed to instantiate the components of a VNF-BD deployment scenario is defined.

6.3.2. Automated Execution

Satisfied all the pre-execution procedures, the automated execution of the Tests specified by the VNF-BD follow:

1. Upon the parsing of a VNF-BD, the Manager must detect the VNF-BD variable input field (e.g., list of resources values) and compose the all the permutations of parameters. For each permutation, the Manager must elaborate a VNF-BD instance. Each VNF-BD instance defines a Test, and it will have its deployment scenario instantiated accordingly. I.e., the Manager must interface an orchestration platform to realize the automated instantiation of each deployment scenario defined by a VNF-BD instance (i.e., a Test). The Manager must iterate through all the VNF-BD instances to finish the whole set of Tests defined by all the permutations of the VNF-BD input fields.

2. Given a VNF-BD instance, the Manager, using the VNF-BD environment settings, must interface an orchestrator platform requesting the deployment of a scenario to realize a Test. To perform such step, The Manager might interface a management function responsible to properly parse the deployment scenario specifications into the orchestration platform interface format.

3. An orchestration platform must deploy the scenario requested by the Manager, assuring the requirements and policies specified on it. In addition, the orchestration platform must acknowledge the deployed scenario to the Manager specifying the management interfaces of the VNF and the other components in the running instances for the benchmarking Test.

4. Agent(s) and Monitor(s) (if existing) and the target VNF must be configured by the Manager according to the components settings defined in the VNF-BD instance. After this step, the whole VNF-BD Test will be ready to be executed.

5. Manager must interface Agent(s) and Monitor(s) (if existing) via management interfaces to require the execution of the benchmarking probers (and listeners, if existing), and retrieve expected metrics captured during or at the end of each Trial. I.e., for a single Test, according to the VNF-BD execution settings, the Manager must guarantee that one or more Trials realize the required measurements to characterize the performance behavior of a VNF.

6. Output measurements from each obtained benchmarking Test, and its possible Trials, must be collected by the Manager, until all the Tests are finished. In the execution settings of the parsed

VNF-BD, the Manager must check the Method repetition, and perform the whole set of VNF-BD Tests (i.e., since step 1), until all methods specified are finished.

7. Collected all measurements from the VNF-BD (Trials, Tests and Methods) execution, the intended metrics, as described in the VNF-BD, must be parsed, extracted and combined to create the corresponding VNF-PP. The combination of used VNF-BD and generated VNF-PP compose the resulting VNF benchmark report (VNF-BR).

#### 6.3.3. Post-Execution

After the process of a VNF-BD execution, some automated procedures, not necessarily mandatory, can be performed to improve the quality and utility of a VNF-BR:

1. Archive the raw output contained in the VNF-PP, perform statistical analysis on it, or train machine learning models with the collected data.

2. Evaluate the analysis output to the detection of any possible cause-effect factors and/or intrinsic correlations in the VNF-BR (e.g., outliers).

3. Review the input VNF-BD and modify it to realize the proper extraction of the target VNF metrics based on the performed research. Iterate in the previous steps until composing a stable and representative VNF-BR.

### 6.4. Particular Cases

As described in [RFC8172], VNF benchmarking might require to change and adapt existing benchmarking methodologies. More specifically, the following cases need to be considered.

#### 6.4.1. Capacity

VNFs are usually deployed inside containers or VMs to build an abstraction layer between physical resources and the resources available to the VNF. According to [RFC8172], it may be more representative to design experiments in a way that the VMs hosting the VNFs are operating at maximum of 50% utilization and split the workload among several VMs, to mitigateside effects of overloaded VMs. Those cases are supported by the presented automation methodologies through VNF-BDs that enable direct control over the resource assignments and topology layouts used for a benchmarking experiment.

### 6.4.2. Isolation

One of the main challenges of NFV is to create isolation between VNFs. Benchmarking the quality of this isolation behavior can be achieved by Agents that take the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the Agent's noisy workload, frequency, virtualization level, among others, must be detailed in the VNF- BD.

### 6.4.3. Failure Handling

Hardware and software components will fail or have errors and thus trigger healing actions of the benchmarked VNFs (self-healing). Benchmarking procedures must also capture the dynamics of this VNF behavior, e.g., if a container or VM restarts because the VNF software crashed. This results in offline periods that must be captured in the benchmarking reports, introducing additional metrics, e.g., max. time-to-heal. The presented concept, with a flexible VNF-PP structure to record arbitrary metrics, enables automation of this case.

### 6.4.4. Elasticity and Flexibility

Having software based network functions and the possibility of a VNF to 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 automated scaling. These state changes and triggers (e.g. the VNF's scaling state) must be captured in the benchmarking results (VNF-PP) to provide a detailed characterization of the VNF's performance behavior in different states.

### 6.4.5. Handling Configurations

As described in [RFC8172], does the sheer number of test conditions and configuration combinations create a challenge for VNF benchmarking. As suggested, machine readable output formats, as they are presented in this document, will allow automated benchmarking procedures to optimize the tested configurations. Approaches for this are, e.g., machine learning-based configuration space sub-sampling methods, such as [Peu-c].

### 6.4.6. White Box VNF

A benchmarking setup must be able to define scenarios with and without monitoring components inside the VNFs and/or the hosting container or VM. If no monitoring solution is available from within the VNFs, the benchmark is following the black-box concept. If, in contrast, those additional sources of information from within the VNF are available, VNF-PPs must be able to handle these additional VNF performance metrics.

## 7. Open Source Reference Implementations

Currently, technical motivating factors in favor of the automation of VNF benchmarking methodologies comprise: (i) the facility to run high-fidelity and commodity traffic generators by software; (ii) the existent means to construct synthetic traffic workloads purely by software (e.g., handcrafted pcap files); (iii) the increasing availability of datasets containing actual sources of production traffic able to be reproduced in benchmarking tests; (iv) the existence of a myriad of automating tools and open interfaces to programmatically manage VNFs; (v) the varied set of orchestration platforms enabling the allocation of resources and instantition of VNFs through automated machineries based on well-defined templates; (vi) the ability to utilize a large tool set of software components to compose pipelines that mathematically analyze benchmarking metrics in automated ways.

In simple terms, network softwarization enables automation. There are two open source reference implementations that are build to automate benchmarking of Virtualized Network Functions (VNFs).

### 7.1. 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).

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 released as open source tool under Apache 2.0 license [gym].

### 7.2. 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 developed 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 aims to be 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 for 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].

## 8. 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.

## 9. IANA Considerations

This document does not require any IANA actions.

## 10. 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).

## 11. References

### 11.1. 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>.

[ETS14e] ETSI, "Report on CI/CD and Devops - ETSI GS NFV TST006 V0.0.9", April 2018, <https://docbox.etsi.org/isg/nfv/open/drafts/TST006_CICD_and_Devops_report>.

[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>.

## 11.2. Informative References

[gym] "Gym Framework Source Code", <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/>.

[Peu-c] M. Peuster, H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for NFV", IEEE/IFIP 14th International Conference on Network and Service Management (CNSM) , 2018, <https://ris.uni-paderborn.de/record/6016>.

[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/