Experiences in IETF-BMWG: Towards a Methodology for VNF Benchmarking Automation

Raphael Vicente Rosa[1], **Christian Esteve Rothenberg**[1]

[1] University of Campinas (UNICAMP) – Brazil

raphaelvrosa@gmail.com,chesteve@dca.fee.unicamp.br

Abstract. *Network Functions Virtualization (NFV) aims at high-end carrier-grade performance but lacks common methodologies for testing Virtual Network Functions (VNFs). Benchmarking VNFs should consider different degrees of freedom instead of the black-box common approaches created for bare metal network functions. We understand such status-quo needs to be altered having basis on the solid ground of extensive and automated experimentation. Since 2015, we have been addressing a role in this scenario, from a position paper to the creation of the draft "Methodology for VNF Benchmarking Automation" in the Internet Engineering Task Force (IETF) Benchmarking Methodology Working Group (BMWG). This paper tells the tale about this draft in BMWG, associated with the perks of developing an open source reference implementation and academic papers, as the means of the old IETF mantra on running code. The story intends to showcase our experiences in IETF and BMWG, covering technical content (e.g., YANG models) as much as draft reviews on mailing-lists.*

1. Introduction

Since its inception in 2012, Network Functions Virtualization (NFV) [Mijumbi et al. 2016] has envisioned carrier grade comparison performance for Virtual Network Function (VNF) embodiments. Throughout the efforts to acquire the knowledge and realize such vision, different approaches have been established towards high performance network functions in a diversity of enabling technologies for VNFs. Hitherto, a myriad of execution environment capabilities have been developed and advanced the state of the art regarding new techniques of specialized packet processing methodologies. Specially in commodity servers, driving forces pushed by new market entrants have supported a major evolution in data plane components that underpin multiple VNF abstractions and implementations (e.g., DPDK, eBPF, XDP).

In reflex of the latest NFV disruptions regarding VNF and execution environment capabilities, some important work [Veitch et al. 2015, Cao et al. 2015, Peuster and Karl 2016] has been performed on the willingness to understand these new degrees of freedom and how they affect VNF performance. Degrees of freedom refer to the associated means to efficiently utilize virtualized resources from compute, storage and network cooperatively. The utilization of those resources might be implemented in diverse settings, such as a programmable data plane, a cross-layer embodiment, and a decoupled control plane.

Nowadays, based on a summary of the academic and industry findings [Van Rossem et al. 2020, Peuster et al. 2019] we understand estimating the behaviour of a VNF depends on three factors:

i. **Internal configuration:** each use case of the VNF might define specific settings for it to work properly, and even each VNF might dispose of specific settings to be configured.
ii. **Hardware and software execution environment:** a myriad of capabilities offered by execution environments might match in a large diversity of manners the possible software arrangements internal of each VNF might be programmable.
iii. **Network workload specificities:** depending on the use case, a VNF might be placed in different scenarios, operating on varied traffic profiles and in demand of a specific performance behavior.

This paper contributes with some reflections on how to tackle the diversity of settings imposed by the above enlisted factors when testing a VNF, in particular benchmarking. In summary, we present a whole set of progressive efforts that among other factors culminated in our participation in the Benchmarking Methodology Working Group (BMWG) at Internet Engineering Task Force (IETF), our related activities realized so far, and our insights on our past efforts and directions for the next steps. We believe telling our story and reflections might bring some clarifications about the "common manners" of participating in IETF.

To construct the baseline knowledge of this paper, we start (Section 2) by establishing a prose on the motivating aspects that lead us to participate in the BMWG in IETF. In Section 3, we depict an overall picture of the content of a draft proposed at BMWG. Hence, we tell the story of Gym (Subsection 3.6), a tool that became our reference implementation for testing VNFs, and how it addresses the methodology proposed by the work-in-progress draft. Later, we discuss (Section 4) the draft content and our personal view on participating in BMWG. Finally, we present some important related work (Section 5) and conclude (Section 6) with some final remarks about the future of our activities in BMWG, and perspectives about testing VNFs.

2. Motivation

Our early idealization of automating the extraction of VNF performance metrics was depicted in a paper in 2015, named VNF Benchmarking as a Service (VBaaS) [Rosa et al. 2015a]. There, not only an initial architecture of a framework was proposed as well as the use cases of the utility of VNF automated benchmarking. One motivating scenario was the ability to quickly fire comparable VNF benchmarks from different vantage points to decide on the most adequate (from a performance perspective) data center PoP and/or NFV Infrastructure (NFVI) providers. Important concepts were derived from this paper giving birth to subsequent work by us and other research groups.

Back there, only a small subset of related work existed involving the formal definition of methods to extract VNF performance metrics. At European Telecommunications Standards Institute (ETSI), the closest related work at NFV Testing (TST) working group, technical specifications mostly addressed the NFV Management and Orchestration (MANO) framework, in specific the NFVI. And the NFV Research Group (NFV-RG), later closed, there were attempts to enlist topics related to NFV research in IETF, where we tried to establish our first contact with IETF. Being welcome in NFV-RG, our work proceeded until the group was terminated and we were recommended to move to BMWG.

Our motivation to propose a draft in BMWG came after we published a paper at IEEE Communications Magazine (Network Testing and Analytics Series) [Rosa et al. 2017], demonstrating the Gym tool, which performs the automated methods to benchmark a VNF. We envisaged formulating a systematized methodology for benchmarking a VNF was in alignment with the BMWG charter and we were developing the tool to perform the "running code" mantra of the IETF. Yet, the other side of the IETF mantra still needed to be conquered towards "rough consensus".

Herein, the terminology "the draft" refers to our work at BMWG entitled Methodology for VNF Benchmarking Automation [Rosa et al. 2019].

3. The Draft: A Methodology for VNF Benchmarking Automation

Nowadays, the draft is moving into its 6th version, but in 2018 it was still named "VNF Benchmarking Methodology". It was not until we built a partnership with another research group interested in benchmarking VNFs that we highly improved the quality of the draft content. Together with Manuel Pesteur and Holger Karl from the Paderborn University we changed the scope of the draft to address the automation aspects of benchmarking a VNF. The draft improved swiftly when the newborn partnership started peer reviewing the draft content with different perspectives about the topic.

After some reviews, nowadays the draft entitled Methodology for VNF Benchmarking Automation, "describes a common methodology for the automated benchmarking of VNFs executed on general-purpose hardware". As stated in the draft content "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".

One of the main manners to improve the draft was to justify its existence at BMWG, thus we defined the reasons to embody automation inside a VNF Benchmarking methodology as: "(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 runtime mechanisms to assist VNF lifecycle orchestration/management workflows, e.g., automated resource dimensioning based on benchmarking insights". Within this new mindset the draft was better aligned with the BMWG charter, in fact being defined as a BMWG milestone.

Most of the content in this section was extracted from the draft itself, to highlight the motivational aspects of the draft and give evidence to some parts of its content.

3.1. A Generic VNF Benchmarking Architectural Framework

In the draft, besides of defining its scope and terminology, there are important aspects that we highlight. In the considerations section of the draft, we disclosure a formal and systemic definition of phases inside of benchmarking procedures. Those provide important support for the whole draft content, and forthcoming work derived from it. The definition of the so called Generic VNF Benchmarking Architectural Framework, see Fig. 1, consists in explaining the roles of each functional component, providing freedom

for implementations to derive their ideas from there, as it enables the automation of VNF benchmarking methodologies.

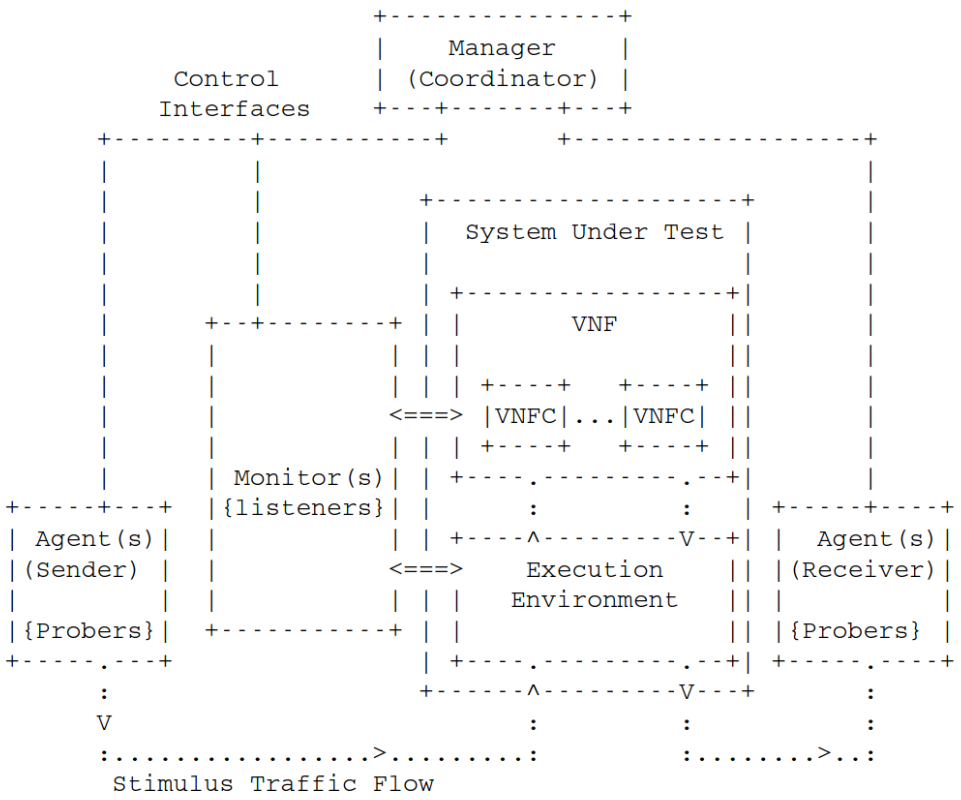

Figure 1. Illustration of the Generic VNF Benchmarking Architectural Framework.

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.

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.

Agent (Active Probing) – executes active stimulus using probers, i.e., benchmarking tools, to benchmark and collect network and system performance metrics.

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.

Monitor (Passive Probing) – 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.

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 oper-

ational parameters, execution of passive monitoring captures, parsing of extracted metrics, and debugging options.

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.

3.2. Deployment Scenarios

A deployment scenario realizes the actual instantiation of physical and/or virtual components of a Generic VNF Benchmarking Architectural Framework needed to enable the execution of an automated VNF benchmarking methodology.

The importance of deployment scenarios relate to the clarification of degrees of freedom the reader of the draft can utilize it to create an actual implementation of a deployment scenario.

The following considerations hold for a deployment scenario:

- Not all components are mandatory for a setup, 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 setup 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.

3.3. Benchmarking Procedures

An 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 topics 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.

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).

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.

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.

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).

3.4. The Automated Benchmarking Methodology

In this section most of the content related to the draft is new, being prepared to be part of its 6th version.

An automated 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).
- **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.
- **Configurability**: Open interfaces and extensible messaging models shall be available between benchmarking components for flexible composition of Test descriptors and platform configurations.
- **Interoperability**: Tests shall be ported to different environments using lightweight components.

To establish the common ground for generic procedures, some concepts were elaborated to conceptualize the flow of information involved in an Automated Benchmarking Methodology.

VNF Benchmarking Descriptor (VNF-BD) – contains all required definitions and requirements to deploy, configure, execute, and reproduce VNF benchmarking tests.

VNF Benchmark Descriptors (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 Benchmark Report (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.

VNF Benchmark Report (VNF-BR) – contains the definition of the inputs and outputs of the automated benchmarking process. Inputs define a VNF-BD, and the variables to be parsed into the VNF-BD and define a benchmarking sample space. The outputs consist of a list of each one of the sampled variables and (i.e., parsed and executed) VNF-BD from the input and its associated VNF Performance Profile (VNF-PP), obtained from the benchmarking process execution of the VNF-BD. In addition, a VNF-BR contains the definitions, if needed, of the orchestrator that realizes the instantiation of the execution environment to enable a VNF-BD deployment scenario.

The workflow, see Fig. 2, of an automated benchmarking methodology consists in defining the inputs (i.e., environment, variables, VNF-BD) of a VNF-BR, submit the VNF-BR to a tool that consumes the inputs of such artifact and elaborates its outputs i.e., a list of entries containing each parsed VNF-BD instance, its associated VNF-PP and the input variables used by them. The list defines in its entries all the possible combinations of the values of the variables contained in the VNF-BR inputs. In details, each one of the variables in the input part of a VNF-BR is defined by: a name (the actual name of the variable); a path (the YANG path of the variable in the input VNF-BD); type (the type of the values, such as string, int, float, etc); class (one of: stimulus, resource, configuration); and values (list of its actual values).

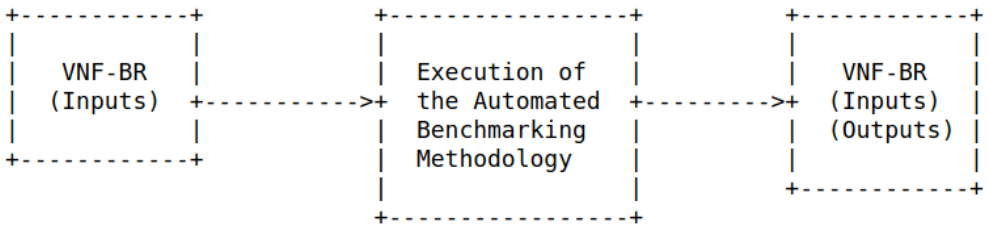

Figure 2. Illustration of the VNF benchmarking process inputs and outputs.

In its essence the execution of an automated benchmarking methodology can be divided in three simple stages: steps before its execution, comprising the planning and the obtaining of software artifacts; the phases of the automated benchmarking procedures; and, the steps after its execution, comprising mostly the analysis and extraction of meaning from the benchmarking results. Each of these stages are defined in the subsections as follows.

3.4.1. Before: Planning

Before the automated benchmarking of a VNF, some procedures must be performed:

1. The VNF target image must be prepared to be benchmarked, having all its capabilities fully described (i.e., model, release, configuration). 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.
2. 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 benchmarking structural and functional settings. At the end of this step, the complete Method of benchmarking the target VNF 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 specified 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.
4. The VNF-BR input variables must be defined, so all the sample space of parameters to be explored by different instances of the input VNF-BD can be generated and instantiated during the automated execution of it.

3.4.2. Automated Execution

The Automated Execution requires the existence of the components depicted in the Generic VNF Benchmarking Architectural Framework, i.e., the proper existence of a VNF-BD scenario.

Given the planning of the automated execution, i.e., the definition of a VNF-BR, a VNF artifact, and an execution environment for the components of the Generic VNF Benchmarking Architectural Framework, the following steps characterize an automated execution of a benchmarking methodology.

1. Given at least the instantiation of a Manager a composed VNF-BR is submitted to it, which triggers the automated execution of the benchmarking methodology defined by the inputs part of the VNF-BR. Manager computes all the combinations of values from the lists of inputs in the VNF-BD, part of the submitted VNF-BR. Each combination of variables define a Test. The VNF-BD submitted serves as a template for each combination of variables. Each parsing of those variables by the VNF-BD template creates a so called VNF-BD instance. 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 parsed input variables. The Manager iterates through the following steps until all the Tests are performed.
2. A Test characterizes by the parsing of one of the sampled combinations of variables, from the inputs part of the VNF-BR, into the VNF-BD, also part of the inputs of the VNF-BR. Two paths might occur, if the VNF-BD scenario needs instantiation, the next step takes place, otherwise the next step is skipped, as the Manager component assumes the VNF-BD scenario is already deployed.
3. The Manager must interface an orchestration engine to realize the automated instantiation of the deployment scenario defined by a VNF-BD instance (i.e., a Test).

To perform such step, The Manager might interface a management function responsible to properly parse the deployment scenario specifications into the orchestration engine interface format. The environment specifications of the VNF-BR provide the guidelines to interface the orchestration engine. The orchestration engine must deploy the scenario requested by the Manager, assuring the requirements and policies specified on it. In addition, the orchestration engine 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. After this step, the whole VNF-BD Test proceedings are ready to be executed.

4. 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 the VNF under test. The number of Trials is defined by each VNF-BD instance.
5. Output measurements from each obtained benchmarking Trials that compose a Test result must be collected by the Manager, until all the Tests are finished. Each set of collected measurements from each VNF-BD (i.e., Trials and Tests) instance must be defined as a VNF-PP by Manager. In this manner, a list of the respective VNF-PP, its associated VNF-BD instance and its input variables compose a list of outputs of the VNF-BR. When all the list of combinations of input variables is explored to obtain the whole list of VNF-BDs and VNF-PPs, the Manager component returns the original VNF-BR submitted to it, including the outputs part properly filled.

3.4.3. After: Analysis

After the automated procedures the analysis of the VNF-BR outputs can be performed to improve the quality and utility:

1. Archive the output VNF-BR from the obtained execution for further utilization and comparability. Perform statistical analysis on the outputs part of the VNF-BR to obtain meaningful significance in intervals of confidence obtained from the repetition of Tests and Trials.
2. Evaluate a root cause analysis on the obtained statistical analysis of the VNF-BR outputs, enabling the the detection of any possible cause-effect factors and/or intrinsic correlations in the VNF-BR outputs (e.g., outliers). Here the applicability of machine learning models can be important to refine the statistical results.
3. According to the analysis of the previous steps, review the input VNF-BD and modify it to realize the proper extraction of the target VNF metrics based on the performed research. Iterate these modifications after the automated execution of the modified VNF-BR until composing a stable and representative VNF-BR.

3.5. YANG Models

In order to create ways to standardize the automated methodology proposed, we needed to define artifacts to express the information flow.s Thus, we introduced YANG models for a VNF-BD[1] and a VNF-PP[2], clarifying their concepts and turning possible a standardize method to implement the ideas of the draft. Later on, a new YANG model was created to represent a VNF-BR, still in the stage to be formalized and added to the draft. Those models define the data structure that establish all the means for any implementation of a Manager component to perform automated benchmarking methodologies. Besides, it was an interesting strategy to produce more attention to the draft, since YANG models became one of the status quo methods to implement ideas of draft and RFCs in IETF.

3.6. A Reference Implementation: Gym

In a draft a major importance is given to it when there exists running code supporting its ideas. The software, named Gym, is a framework for the automated benchmarking of VNFs, coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa et al. 2015a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFV-RG and BMWG. Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas, such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, etc).

In [Rosa et al. 2017], Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. In [Rosa and Rothenberg 2017], 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 and contributed with discussions of the lessons learned and the overall NFV testing landscape, including automation. Gym stands as one open source reference implementation that realizes the VNF benchmarking methodology presented by the draft[3]. Gym is released as open source tool under Apache 2.0 license[4].

3.6.1. Gym Architecture

One of the major points of Gym's architecture sits in the fact is was built by the draft at the same time it also influenced the draft content. Those characteristics make possible back and forth reviews of both the draft as much as the Gym source code simultaneously. For instance, the definition of the draft concepts about probers and listerners derived from the Gym architecture, while the utilization of YANG models by Gym was something new brought by the draft.

In essence Gym is faithful to the draft in what concerns the definition and utilization of YANG models, however its internal architecture components differ from the draft.

[1]https://github.com/raphaelvrosa/vnf-bench-model/tree/master/vnf-br/yang/vnfbd

[2]https://github.com/raphaelvrosa/vnf-bench-model/tree/master/vnf-br/yang/vnfpp

[3]https://github.com/raphaelvrosa/vnf-bench-model/blob/master/experiments/gym-bench-02/results/analysis.ipynb

[4]https://github.com/intrig-unicamp/gym

Such freedom of choices is something important that was realized while designing the Generic VNF Benchmarking Architectural Framework. In Gym, the Manager component is divided into two components, a Manager and a Player. In summary, it conceives a decoupling of functions, Player performs the execution of Tests while Manager performs the execution of Trials. This design choice created a modular architecture for Gym, facilitating its implementation and utilization. Besides that, Gym implements Agent and Monitor components as defined by the draft.

3.6.2. Implementation Details

Each component in Gym is a micro-service, communicating with each other via gRPC[5] interfaces and protobuf[6] messages. All the components are written in the Python programming language, version 3.8. The support of YANG methods was provided thanks to the pyangbind python package, which enables the compiling of a YANG model into a python source code, which is used as an API by Gym when utilizing VNF-BR, VNF-BD and VNF-PP models. In Gym the workflow of execution of a VNF-BR takes place in the following manner:

1. Player parses a submitted VNF-BR, defines the set of VNF-BD instances by multiplexing all the input variables of the VNF-BR. It might interface different plugins to perform instantiation of the deployment scenario by an orchestration engine. Performed the scenario instantiation, Player reaches the Manager component with a Task message, containing the set of Trials to be executed in a realization of a VNF-BD instance Test.
2. Manager receives the Task and properly deliver a set of Instruction messages to Agent and Monitor components, containing Actions that trigger each of the probers or listeners to be used for the extraction of VNF metrics. Manager repeats this set of instructions according to the number of Trials. After those are completed, Manager returns a Report message to Player signaling the Test ended.
3. Player iterates throughout all the Tests of each VNF-BD instance, until finishing the whole combination of the variables in the inputs of the VNF-BR. Each VNF-BD instance has a VNF-PP generated as the Test metrics are extracted. Finally, Player creates the outputs part of the VNF-BR and returns it through the interface it was submitted.

4. Discussion: Evolving a draft in BMWG

One of the major goals of this paper consists in discussing the lessons learned from trying to evolve a draft in BMWG, therefore besides exposing the draft content as done so far, in this section we bring important insights that led us until here.

The perks of creating an automated benchmarking methodology for VNFs: As much as the BMWG charter scopes the benchmarking of VNFs, we believe there exist issues that still the working group needs to evolve related to that topic: the definition of a common terminology; the clarification of boundaries with ETSI NFV; the consolidation of the scope of automation in BMWG. As the draft tries to bring novelty to a large sum of each

[5]https://grpc.io/

[6]https://developers.google.com/protocol-buffers/

of these topics into the working group, there exists many doubts on how broad its scope might span at the point of trying to creating a unique methodology for all VNFs. Those factors conceive the biggest challenges of the draft in BMWG.

A view on YANG models: It appears YANG models attract the attention of readers in IETF, bringing a degree of seriousness to the draft. Still, while designing a YANG model a series of questions emerge, and the proper adequacy to IETF standards seems difficult to attain (included adding the YANG models to the draft), but fluency appears to eventually become common after many iterations and errors. One of the important aspects in this journey is creating usable and testable models by a reference implementation. From another point of view, YANG was defined as a language for data models, targeting specific configurations. One of our challenges is adapting YANG to fit the varied settings and metrics needed to be expressed in the VNF-BD and VNF-PP models.

Running code and test cases in building a reference implementation: Clearly, the running code aspects of building a draft in IETF gets increasing difficult as the draft evolves and new ideas are included into it. Fast cycles of iteration in the source code of the artefacts that realize the draft must happen as continuously as possible. This approach creates the means to validate the draft content and showcase its importance into actual use cases. In this scope, publishing academic papers and demonstrations sets a strategy detaining some impact in the support of the draft in BMWG.

Overall IETF/BMWG impressions: peer reviewing is an important aspect of any working group in IETF, this is how actually a draft might evolve until becoming a RFC. The rough consensus in this case needs to perpetuate in the mailing-list as much as in the meetings, as the members provide support for the draft, increasing its chances to be adopted and evolved. BMWG, one of the oldest but not so popular IETF working groups nowadays, detains its own way of handling drafts, with few expert participants and concise RFCs. Adapting to this *modus operandi* is not trivial, and still is one of the major challenges we have been facing towards a methodology for VNF benchmarking automation.

Implications of the draft in academy and industry: originally the subject about VNF benchmarking emerged from the industry and gained adoption into the academic community. Thenceforth, tools and methodologies for the analysis of the benchmarking results were elaborated. The difficulty here lays in the fact those related work could not be easily comparable or repeatable, and even not extended. The draft as much as its reference implementation comes to fill this gap. In addition it aims to create a data repository of extracted VNF-BR, enabling analytics research on them.

5. Related Work

As previously stated, in the beginning, there was almost no related work on whatsoever called an automation of a benchmarking methodology for VNFs. During the evolution of the draft and work on academic papers, different related work emerged into the definition of benchmarking methods for VNFs and its analysis for the purposes of orchestration.

By prototyping Gym [Rosa et al. 2015b, Rosa et al. 2016], we aimed alignment with previous BMWG work on "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure" [Morton 2016], taking guidelines towards VNF benchmarking and standardization. Likewise, Gym was influenced by related efforts at the

ETSI ISG NFV Testing Group [ETSI GS NFV-TST 2016] defining requirements and recommendations for VNFs and NFVI validation. Closest related to Gym, OPNFV incubated projects include *(i)* Yardstick[7], targeting infrastructure verification; *(ii)* Bottlenecks[8], proposing a framework to execute automatic methods of benchmarks to validate VNFs deployment during staging; and *(iii)* VSperf project[9], targeting test-customizations (e.g., software components, load generators) being suitable for different switching technologies in a telco NFV environment, with experiences described in [Tahhan et al. 2016]. Compared to Gym, these efforts are very much tied to their choice of technologies, compromising portability and repeatability due to the focus on supporting OPNFV developments without broader aspirations of generic VNF testing tools.

6. Conclusions

The path of VNF testing (dimensioning, benchmarking, verification) sits unclear, as a myriad of enabling technologies for NFV keeps emerging. Added complexity exists specially when more and more such technologies tune VNFs in ways that: decouple the network functions control and data planes; define VNFs coded as layered/distributed software components in a cloud mesh environment; and utilize multiple hardware features (e.g., Enhanced Platform Awareness). Therefore, our view on the extraction of a VNF performance profile culminates into automation, due to the huge sample space of variables involved in resource allocation, VNF configuration, and their stimuli traffic profiles.

The roadmap for defining a methodology for VNF benchmarking automation already started in BMWG, in a sinuous path we proposed a draft by which has its story told in this paper. The design choices we envisage for the draft were not always clear, however guided by academic publications and insights from running code. Through our collaboration with other researchers, we accomplished important milestones while evolving the draft, such as the construction of YANG models and comparative implementations. Still, there are challenges towards making those ideas understandable in BMWG to evolve the draft eventually into a possible RFC. Hitherto, our efforts so far have inspired relevant related work. Therefore, we aim to push forward our work at BMWG, more than ever looking for the manners to gain support and showcase demonstrations of running code towards the needed rough consensus.

7. Acknowledgments

This work was partially funded by Ericsson projects UNI.62, UNI.64 and UNI.67. The writing of this paper happened while Dr. Rosa was partially funded by the Electricity Sector Research and Development Program PD-00063-3058/2019 - PA3058: ”MERGE - Microgrids for Efficient, Reliable and Greener Energy”, regulated by the National Electricity Agency (ANEEL in Portuguese), in partnership with CPFL Energia (Local Electricity Distributor). The views expressed are solely those of the authors and do not necessary represent any official standpoint.

[7]https://wiki.opnfv.org/display/yardstick - Accessed on 2020-09-10

[8]https://wiki.opnfv.org/display/bottlenecks - Accessed on 2020-09-10

[9]https://wiki.opnfv.org/display/vsperf/VSperf+Home - Accessed on 2020-09-10

References

Cao, L., Sharma, P., Fahmy, S., and Saxena, V. (2015). NFV-VITAL: A framework for characterizing the performance of virtual network functions. In *IEEE Conference on NFV-SDN*, pages 93–99.

ETSI GS NFV-TST (2016). ETSI GS NFV-TST 002 V1.1.1 - Report on NFV Interoperability Testing Methodology. Accessed on 2017-06-01.

Mijumbi, R. et al. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

Morton, A. (2016). Considerations for benchmarking virtual network functions and their infrastructure. Internet draft. Accessed on 2017-06-01.

Peuster, M. and Karl, H. (2016). Understand your chains: Towards performance profile-based network service management. In *Proceeding of the Fifth European Workshop on Software Defined Networks (EWSDN)*.

Peuster, M., Schneider, S., Zhao, M., Xilouris, G., Trakadas, P., Vicens, F., Tavernier, W., Soenen, T., Vilalta, R., Andreou, G., Kyriazis, D., and Karl, H. (2019). Introducing automated verification and validation for virtualized network functions and services. *IEEE Communications Magazine*, 57(5):96–102.

Rosa, R. V., Bertoldo, C., and Rothenberg, C. E. (2017). Take your vnf to the gym: A testing framework for automated nfv performance benchmarking. *IEEE Communications Magazine*, 55(9):110–117.

Rosa, R. V. and Rothenberg, C. (2017). Taking Open vSwitch to the Gym: An Automated Benchmarking Approach. *In IV Workshop pre IETF/IRTF*.

Rosa, R. V., Rothenberg, C. E., Peuster, M., and Karl, H. (2019). Methodology for VNF Benchmarking Automation. Internet-Draft draft-rosa-bmwg-vnfbench-05, Internet Engineering Task Force. Work in Progress.

Rosa, R. V., Rothenberg, C. E., and Szabo, R. (2015a). VBaaS: VNF Benchmark-as-a-Service. In *2015 Fourth European Workshop on Software Defined Networks*, pages 79–84.

Rosa, R. V., Rothenberg, C. E., and Szabo, R. (2015b). VNF Benchmark-as-a-Service. Internet draft. Accessed on 2017-06-01.

Rosa, R. V., Rothenberg, C. E., and Szabo, R. (2016). VNF Benchmarking Methodology. Internet draft. Accessed on 2017-06-01.

Tahhan, M., O'Mahony, B., and Morton, A. (2016). Benchmarking virtual switches in opnfv. IETF BMWG: Internet draft.

Van Rossem, S., Tavernier, W., Colle, D., Pickavet, M., and Demeester, P. (2020). Vnf performance modelling: From stand-alone to chained topologies. *Computer Networks*, 181:107428.

Veitch, P., McGrath, M. J., and Bayon, V. (2015). An Instrumentation and Analytics Framework for Optimal and Robust NFV Deployment. *IEEE Communications Magazine*, 53(2):126–133.