NFVRG
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

R. Rosa
C. Rothenberg
Unicamp
R. Szabo
Ericsson

October 19, 2015

VNF Benchmark-as-a-Service
draft-rorosz-nfvrg-vbaas-00

Abstract

Network Function Virtualization (NFV) promises the delivery of flexible chaining of Virtualized Network Functions (VNFs) with portability, elasticity, guaranteed performance, reliability among others. Service providers expect similar behavior from NFV as their current appliances based infrastructure, however, with the ease and flexibility of operation and cost effectiveness. Managing for predictable performance in virtualized environments is far from straightforward. Considering resource provisioning, selection of an NFV Infrastructure Point of Presence to host a VNF or allocation of resources (e.g., virtual CPUs, memory) needs to be done over virtualized (abstracted and simplified) resource views. In order to support orchestration decisions, we argue that a framework for VNF benchmarking is need.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Introduction

Network Function Virtualization (NFV) pursues delivering Virtualized Network Functions (VNFs) in NFV Infrastructure (NFVI) where requirements of portability, performance, elasticity, among others, are demanded by carriers and network operators [ETS14a]. Particularly, in the short term, performance and elasticity are important factors to realize NFV concepts [BVLS15]. Associated to these goals VNFs need to be deployed with predictable performance, taking into account the variable processing overheads of virtualization and the underlying available NFVI resources.

In essence, NFV Management and Orchestration (MANO) plane manages infrastructure and network services resources [ETS14c]. This involves taking into account performance considerations for VNF-FGs allocation (e.g., fulfilling NFVI resources by orchestration demands). Furthermore, when attempting to manage service assurance of embedded VNF-FGs [PSPL15] MANO tasks may depend on analytics

measurements for scaling and migration purposes. As a consequence Network Function Virtualization Orchestrator (NFVO) needs a coherent understanding of performance vs. resource needs for VNFs specific to NFVI Points of Presence (PoPs).

In what follows we motivate the need for a VNF benchmarking, define related terms and introduce a framework proposal.

2. Terms and Definitions

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).

NFVO: NFV Orchestrator - functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (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.

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 lifecycle of a VNF instance.

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

MANO: Management and Orchestration - In the ETSI NFV framework [ETS14c], this is the global entity responsible for management and orchestration of NFV lifecycle.

VBaaS: VNF Benchmark-as-a-Service - is a VNF Profile extraction service of a Virtualized Infrastructure Manager (VIM) overseeing an NFVI PoP.

VNF-BP: VNF Benchmarking Profile - the specification how to measure a VNF Profile. VNF-BP may be specific to a VNF or applicable to several VNF types. The specification includes structural and functional instructions, and variable parameters (metrics) at different abstractions (e.g., vCPU, memory, bandwidth, delay; session, transaction, tenants, etc.).

VNF Profile: is a mapping between virtualized resources (e.g., vCPU, memory) and VNF performance (e.g., bandwidth, delay between in/out or ports) at a given NFVI PoP. An orchestration function can use the VNF Profile to select host for a VNF and to allocate the desired resources to deliver a given (predictable) service quality.

3. Motivation

Let us take an example where a VNF Developer creates a new code base for a VNF. She is able to optimize her VNF for multiple execution environments and offers these as a set of different but equivalent implementations of VNF1 for Service Providers (SPs) (see Figure 1). Orchestration managers (e.g., NFVO) must know adequate infrastructure resources to allocate when deploying VNF1, specially if it belongs to a network service (VNF-FG) with a target SLA, e.g., min throughput or max latency. However, instances of VNF1 optimized for heterogeneous NFVI PoPs may need different resources/settings for the same behavior (performance). Orchestrator (NFVO) needs to take into account all these requirements to embed VNFs and allocate proper infrastructure resources in accordance with network service intents (SLAs).

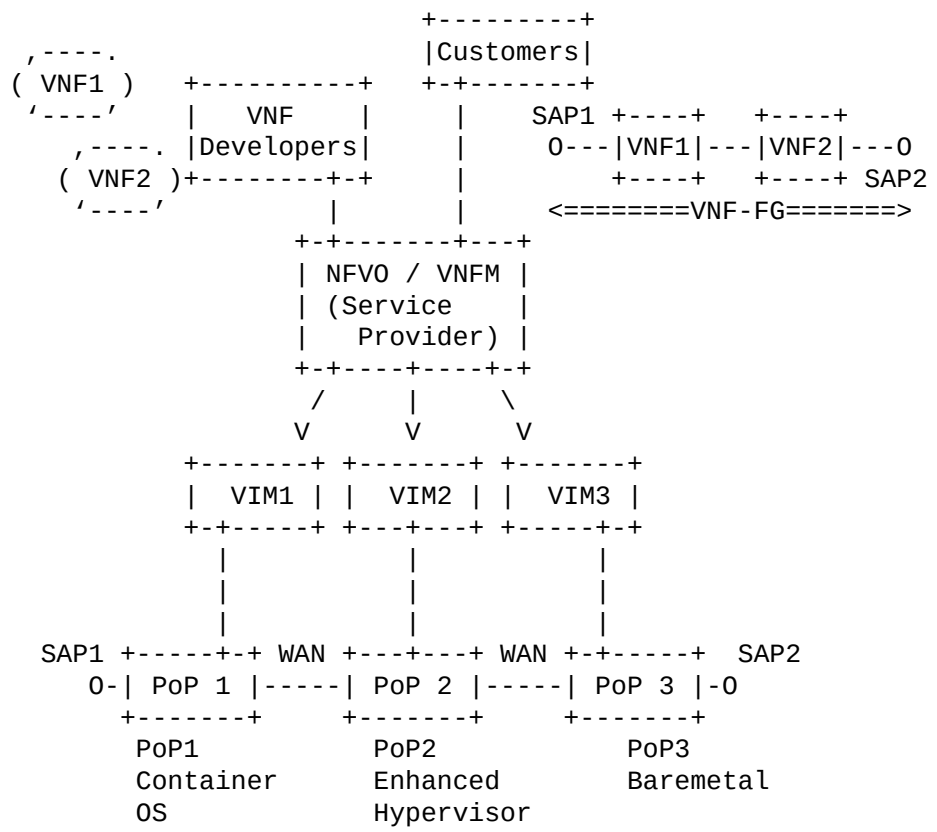

Figure 1: NFV MANO: Customers and VNF Developers

4. Problem Statement and Challenges

Previous section motivates NFVO needs of knowledge about correlations between VNFs performance and NFVI PoP resources.

Suppose that specifications, based on developers' definitions, exist in VNF Descriptors [ETS14d] describing performance associated to specific NFVI PoP resources. Such VNF, when deployed, may present oscillations in its performance because of changes in the underlying infrastructure (due to varying customers workloads, traffic engineering, scaling/migration, etc). Therefore, we believe we need descriptions on how to benchmark VNFs' resources and performance characteristics (e.g., VNF Profiles) on demand and in a flexible process supporting different NFVI PoPs, which could create service level specification associations.

Some of the associated challenges are:

Consistency: Will the VNF deployed at a NVFI PoP deliver the performance given in the VNF Profile for the NFVI PoP under different workloads? The performance of a VNF may not only depend on the resources allocated to the VNF but on the overall workload at the target NFVI PoP.

Stability: Benchmark measurements need to present consistent results when applied over different NFVI PoPs. How express benchmarking descriptions to fit in different virtualized environments? How to uniformly handle service/resource definitions and metrics?

Goodness: A given VIM / NFVI PoP may virtualize (i.e., hide) certain resource details; hence a benchmark evaluation might be executed in a different resource set of the NFVI PoP unlike the final production environment. How good benchmark results can correspond to actual workloads in virtualized environments?

VNF benchmarking service description: How to define the VNF benchmarking process at the abstraction of the corresponding component (e.g., VIM)? How to offer benchmarking as a service to be consumed by the different stakeholders?

VNF benchmarking versus monitoring: How can a VNF benchmarking process complement continuous monitoring of NFVI? When does a VNF benchmarking process surpass in performance and costs the execution of continuous monitoring process? And for which type of NFVI PoPs and/or VNFs is it necessary/sufficient?

5. Proposed Approach

Definition: VNF Benchmarking as a Service (VBaaS) -- a standalone service that can be hosted by orchestration managers (e.g., NFVO) to report a VNF Profile. VBaas might take inputs consisting of VNF-FGs determining VNFs to be evaluated under specific NFVI targets. As output, VBaaS returns a Service Graph based on the VNF-Benchmark Profile (VNF-BP) containing different parameters used to evaluate all required candidates (PoPs) for the specified VNF. Such interactions are abstracted by the VBaaS API. A VBaaS Service Graph is deployed by orchestration managers (NFVO), VBaaS receives benchmarking results and builds VNF Profiles. VNF performance values or VNF resource values can be provided, and VBaaS can only complement the VNF-FG missing parts to build VNF-BPs, or report an existing up-to-date VNF Profile for pre-defined values.

In comparison with Figure 1, the VBaaS approach introduces two information bases, a VBaaS component and an API to interact with NFVO, as shown in Figure 2. On one hand, the Network Function

Information Base (NF-IB) holds the VNF Profiles and on the other hand the VBaaS-IB holds the VNF Benchmark Profiles (VNF-BPs).

Definition: VNF Profile -- is a mapping between virtualized resources of a certain NFVI PoP (e.g., virtual machine with vCPU, memory) and VNF performance (e.g., throughput, delay, packet loss between in/out or ports).

An orchestration function can use the VNF Profile to select hosts for VNFs in a VNF-FG and to allocate the necessary resources to deliver a given (predictable) service quality (SLA).

Figure 2 shows an example VNF1 and VNF2, which are implemented by a VNF Developer, who provides metrics for VBaaS build VNF-BPs. The VNF-BP is then used by VBaaS when NFVO needs to evaluate the available NFVI-PoPs for different resources configuration. In this case, NFVO instantiates the VNF-FG (VBaaS Service Graph) as defined in VNF1-BP; configures the variable parameters in the configuration template, e.g., CPU and memory values and handles the configuration file to the measurement controller (part of the VNF-FG definition) to execute the benchmarking. The measurement controller make the measurement agents execute the benchmark tests (the execution environment might also be monitored by the agents) and reports the results back to the VBaaS. As a result, resource and performance associations for the VNF is created for the evaluated the NFVI PoPs into the NF-IB. Figure 2 shows, as an example, that VNF1 needs 2CPU (cores) and 8GByte of memory to handle up 10Mbps input rate with transaction delay less than 200ms at NFVI PoP1.

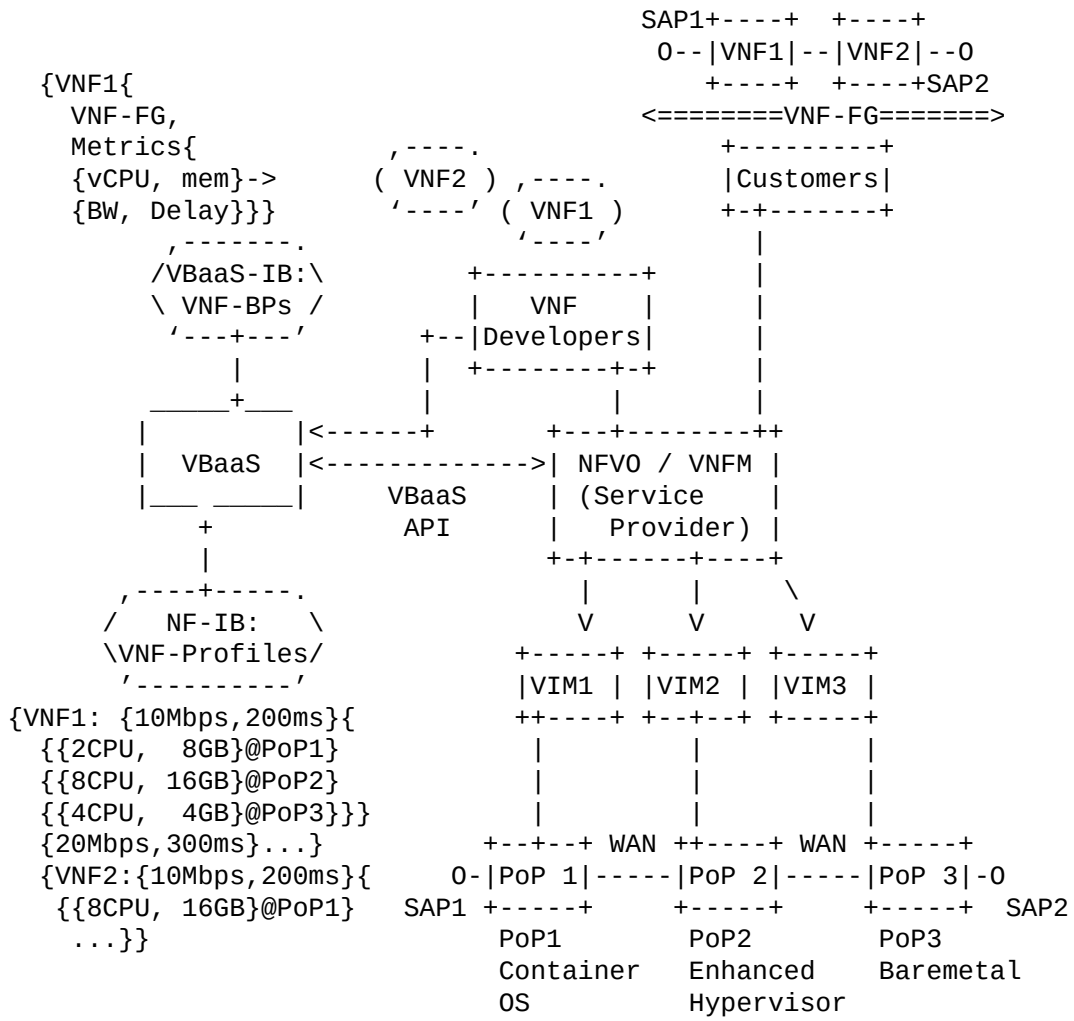

Figure 2: NFV MANO and VBaaS Approach

On the left side of the NFVO in Figure 2 is the resulting NF-IB by the application of VBaaS-IB profiles on the extraction of VNFs performance characteristics in different NFVI PoPs. Basically, a VNF profile corresponds to performance metrics in different environments. For example, VNF1 to offer 10Mbps bandwidth with 200ms latency needs (if deployed) in PoP1 2CPU (cores) and 8GB of memory, meanwhile in PoP2 requires 8CPU and 16GB. Similar results can be presented for VNF2, if catalogued as a VNF profile. In this scenario, VBaaS can be seen featuring VNFs in different PoPs in an on-demand composition of profiles, for example, to be used in provisioning VNF-FGs containing VNF1 or VNF2, attending PoPs 1, 2 or 3 resources consumption. Such profiles present performance metrics of VNFs in different NFVI PoPs

when, for example, NFVOs need to deploy particular VNF-FGs that contain elements already certified in NF-IB profiles.

Definition: VNF Benchmarking Profile (VNF-BPs) -- the specification of how to measure the VNF Profile for a given VNF. The specification includes structural (e.g., measurement components defined as a VNF-FG) and functional (e.g., measurement process definitions and configurable parameters) instructions, and variable parameters (metrics) at different abstractions (e.g., vCPU, memory, bandwidth, delay; session, transaction, tenants, etc.).

Note: a VNF-BP may be specific to a VNF or applicable to several VNF types.

Figure 3 presents details of VBaaS API regarding the interactions with NFVO and the construction of VBaaS Service Graphs based on VNF-BPs. As a VNF-FG (deployment request), NFVO demands VNF1 allocated to a specific PoP1 and the remaining virtualized view of the current resources for the VNF-FG deployment (e.g., interconnected PoPs). VBaaS (Manager Service) looks for VNF1-BP in VBaaS-IB, complements the requested VNF-FG with benchmark instances (described below) to perform evaluations and monitor resources in VNF1 and PoP1. NFVO receiving a VNF-FG reply (deployment order of a VBaaS Service Graph) instantiates it as a normal provisioning process of network services. Benchmark reports of the deployed VNF-FG are sent to VBaaS and it defines the construction of the VNF-Profile associated with that evaluation.

We believe in the existence of VBaaS instances (VNFs) which compose a VBaaS Service Graph to perform probing and monitoring activities on selected targets (VNF in a specific NFVI PoP). Those are defined as Manager, Monitor and Agent, described in details below.

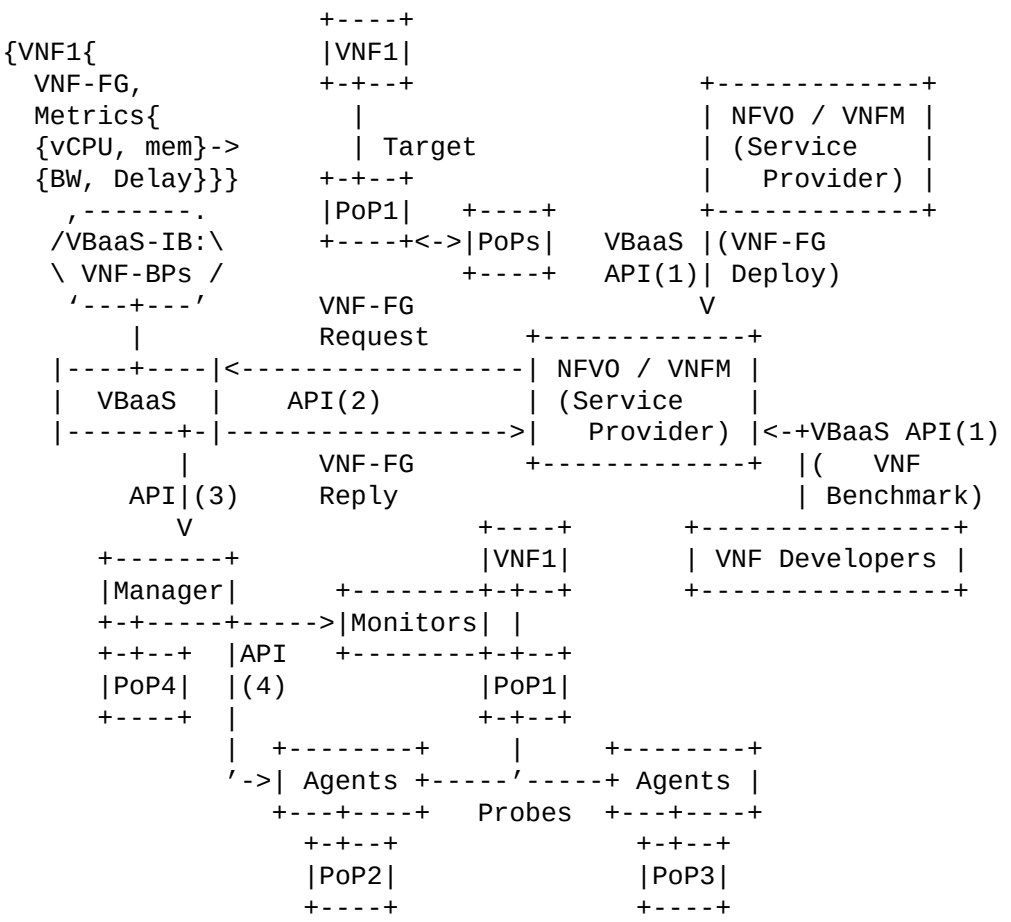

Figure 3: VBaaS APIs

Agent -- executes active probes using benchmark tools to collect network and system performance metrics by interacting with other agents. While a single Agent is capable of performing localized benchmarks (e.g., Stress tests on CPU, memory, I/O), the interaction among distributed agents enable the generation and collection of end-to-end metrics (e.g., packet loss, delay). Consider here the possibility of one end be the VNF itself where, for example, one-way packet delay is evaluated. For instance, it can be used for probe measurements of throughput and latency in a multi-point distributed topology. Agent's APIs are open and extensible (e.g., to plug-and-bench new tools and metrics) and receive configuration and run-time commands from the Manager for synchronization purposes (e.g., test duration/repetition) with other Agent and Monitor instances.

Monitor -- when possible it is placed inside the target VNF and NFVI PoP to perform active and passive monitoring and metric collection based on benchmarks evaluated according to Agents' workloads. For instance, to monitor CPU utilization of NFVI PoP considering packet length traffic variations sent by Agents. 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. Monitors can plug-and-bench new tools/metrics and maintain an interface with Manager to synchronize its activities with Agents. Besides, they can be instantiated, if allowed, inside the target VNF (e.g., as a plug-in process in a virtualized environment) to check internal behaviours, alongside the VNF deployment environment, i.e., NFVI PoP, or in both places.

Manager -- is responsible for (i) the coordination and synchronization of activities between agents and monitors, (ii) collecting all benchmark raw results, and (iii) aggregating the inputs to construct a profile report that correlates different metrics as required by the VNF-BP. Therefore, it executes the main configuration, operation and management actions to deliver the results as specified by NFVI PoPs or VNF-BPs (e.g., instantiation of agents and monitors activities along- side tools and metrics configuration). Manager uses APIs to read and set configuration parameters for benchmarking instances such as metric parameters (e.g., bandwidth or packet loss probes) according to the VBaaS process. APIs are also used to receive benchmark instructions from VBaaS and send the reports of finished benchmark procedures.

The benchmarking process, however, can be offered as a service (see API (1) in right side of Figure 3). For example, a NFVO may be able to handle a VBaaS-IB and offer a VNF Benchmarking as a Service to its client, e.g., another NFVO. Similarly, a second NFVO may store benchmark measurements in its NF-IB so it can report it to multiple clients if it shares resources among other NFVOs. Alternatively, an NFVO may offer to it's developers the VNF Benchmarking as a Service, to assess the performance of a VNF in the operator's environment. This may be part of a DevOps primitive (e.g., VNFs continuous integration).

6. Use Case: Unify Modelling

Unify looks for a generic description of compute and network resources as a yang model named virtualizer [Szab15]. It allows the aggregation of resources (e.g., nodes capabilities and NF instances) and their abstractions in a structure nicknamed Big Switch with Big Software (BiS-BiS) (joint software and network resource abstraction

with a joint control API), which allows recursive control plane structuring for multi-domain hierarchies. For example, supported network functions of a joint compute and network virtualization (a BiS-BiS node) can be described with their associated resources and performance mapping, e.g., {cpu, memory, storage} -> {bandwidth, delay} (see Figure 5 based on virtualizer xml example in [Szab15]).

In this example, the interface between NFVOs (Producer and Consumer of VBaaS API (1)), as shown in Figure 3, is described by the UNIFY virtualization model [Szab15]. Figure 4 shows an example netconf message exchange over the domain virtualization model. First the consumer reads in (1) and (2) the virtualization view, which includes the supported NFs reported in the capabilities section. Next in (3) the consumer requests a capability report of NF1 at a given virtualized node (UUID001), by defining the required delay and bandwidth performance parameters of the NF1 (see Figure 5. This is a dimensioning request according to [ETS14t], i.e., the VBaaS service must evaluate and provide cpu, memory and storage values at the given virtualized node to deliver the given network performance characteristics. The start of the VBaaS process is acknowledged by a (4) rpc-ok. Once the VBaaS process completes a notification is sent (5) to the consumer, which reads in (6) and (7) the updates to the NF1 capability report, i.e., the missing cpu, memory and storage parameters (see Figure 6).

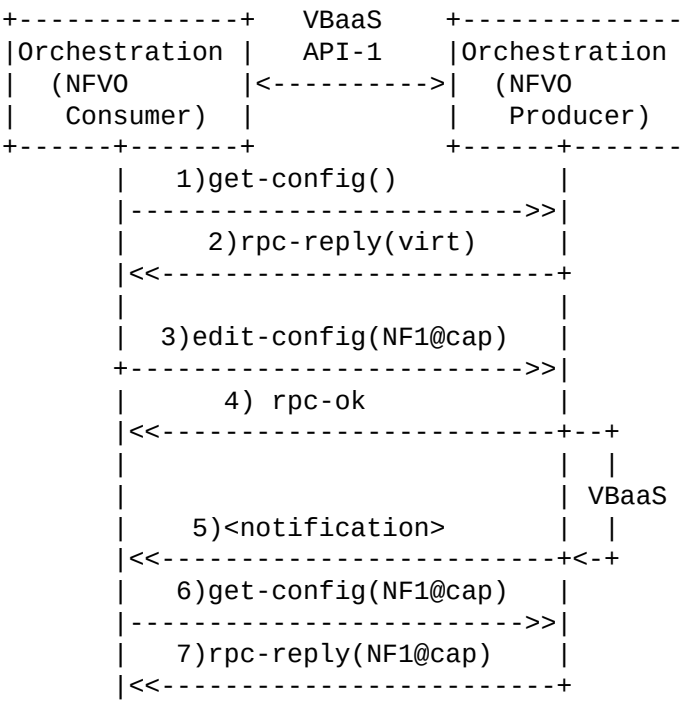

Figure 4: Sequence Diagram of VBaaS API (1) - Multi-Domain Interaction

```
<virtualizer xmlns="http://fp7-unify.eu/framework/virtualizer">
    <id>UUID001</id>
    <name>Single node simple VBaaS request</name>
    <nodes>
        <node>
            <id>UUID11</id>
            <name>single Bis-Bis node</name>
            <type>BisBis</type>
            <ports>
                ...
            </ports>
            <resources>
                <cpu>12</cpu>
                <mem>64 GB</mem>
                <storage>20 TB</storage>
            </resources>
            <capabilities>
              <supported_NFs>
                      <node>
                          <id>NF1</id>
                          <name>vFirewall</name>
```

```
                            <type>Stateful virtual firewall C</type>
                            <ports>
                                <port>
                                    <id>1</id>
                                    <name>in</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                                <port>
                                    <id>2</id>
                                    <name>out</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                            </ports>
                            <resources>
                                        <cpu> ? </cpu>
                                        <mem> ? </mem>
                                        <storage> ? </storage>
                                    </resources>
                        <links>
                                        <link>
                                            <id>int0</id>
                                            <name>internal horizontal</name>
                                            <src>../../ports/port[id=1]</src>
                                            <dst>../../ports/port[id=2]</dst>
                                            <resources>
                                                <delay> 20 us </delay>
                                                <bandwidth> 10 GB </bandwidth>
                                            </resources>
                                        </link>
                                    </links>
                        </node>
                  </supported_NFs>
               </capabilities>
          </node>
      </nodes>
  </virtualizer>
```

Figure 5: VBaaS_Request(NF-FG-1)

```
  <virtualizer xmlns="http://fp7-unify.eu/framework/virtualizer">
      <id>UUID001</id>
      <name>Single node simple VBaaS report</name>
      <nodes>
          <node>
```

```
              <id>UUID11</id>
              <name>single Bis-Bis node</name>
              <type>BisBis</type>
              <capabilities>
                <supported_NFs>
                        <node>
                            <id>NF1</id>
                            <name>vFirewall</name>
                            <type>Stateful virtual firewall C</type>
                            <ports>
                                <port>
                                    <id>1</id>
                                    <name>in</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                                <port>
                                    <id>2</id>
                                    <name>out</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                            </ports>
                            <resources>
                                        <cpu>2</cpu>
                                        <mem>8 GB</mem>
                                        <storage>20 GB</storage>
                                    </resources>
                        <links>
                                        <link>
                                            <id>int0</id>
                                            <name>internal horizontal</name>
                                            <src>../../ports/port[id=1]</src>
                                            <dst>../../ports/port[id=2]</dst>
                                            <resources>
                                                <delay> 20 us </delay>
                                                <bandwidth> 10 GB </bandwidth>
                                            </resources>
                                        </link>
                                </links>
                        </node>
                </supported_NFs>
              </capabilities>
          </node>
      </nodes>
  </virtualizer>
```

Figure 6: VBaaS_Request(NF-FG-2)

Following the above example and the terminology of [ETS14t], the different tasks of performance verification, benchmarking and dimension can be mapped as follows:

Verification: Both {cpu, memory, storage} and {delay, bandwidth} parameters are provided and the VBaaS system verifies if the given association is correct or not. In the case of a failure the entry might be removed or updated with correct values.

Benchmarking: Where {cpu, memory, storage} parameters are provided and the VBaaS service fills-in the corresponding {delay, bandwidth} performance parameters. Note, such request might create more entries, like an entry with minimal delay and an entry with maximum bandwidth.

Dimensioning: Where {delay, bandwidth} performance parameters are provided and the VBaaS service fills-in the corresponding {cpu, memory, storage} resource parameters (as shown in the above example).

7. Related drafts and open source projects

Definetly, VBaaS intersects IETF Benchmarking Methodology Work Group (BMWG) goals. For example, in [Mor15], "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure" presents relevant statements concerning, for example, % of utilization of NFVI PoPs resources; noisy behavior caused by VNFs operations and misbehavior in NFVI PoPs; long-term tests which represent service continuity guarantees for VNFs; and considerations on how shared resources dependencies may affect consistent benchmark results. All these NFV benchmarking aspects are relevant for VBaaS, as well other current [Tah15] and future recommendations from BMWG.

The main DevOps Principles for Software Defined Infrastructures (SDIs) (e.g., NFV and SDN), well explained in [Meir15], are associated with the main design concepts of VBaaS. By definition, DevOps principles look for deployment with repeatable and reliable processes (VNF-BPs), as well modular, e.g., VNF Profiles assisting orchestration decisions. Besides, it also stands for development and testing procedures to assist VNF Developers (VBaaS API) specify modular applications/components (VNF-BPs) against production-like systems. In addition, DevOps intends to define ways of monitor and validate operational quality of services before and after services deployment, e.g., respectively applying VNF-BPs in multiple NFVI PoPs and over instantiated VNFs. All these aspects illustrate how VBaaS can help in amplifying the cycle between VNFs continuous changes of

development and Operators desires for infrastructure stability and reliability, therefore, helping the consolidation of DevOps principles in SDIs.

In OPNFV (Open Platform for NFV) community, projects dedicated to VIM and NFVI tests are being developed, such as vSwitchPerf, Pharos, Functest, among others. Specifically, Yardstick [Yard15] is closely related with VBaaS, because it stands for a common definition of tests to verify the infrastructure compliance when running VNF applications. While VBaaS is still building its concepts, Yardstick already defined requirements, methodologies (including metrics for compute, network and storage domains), besides functional code. The specification of yaml files to define structures like scenarios, contexts and runners, as a Yardstick nomenclature to compose tests, represents associations with VNF-BPs concepts. As Yardstick future work intends to apply tests in compute and storage domains, and store tests results, it is on its way the possible intersection with VBaaS concepts, like VNF Profiles.

T-NOVA [TNOVA15] pursues the definition of a marketplace for VNFs (and turn them instanced as a service). Regarding the description of capabilities for the allocation of VNFs, T-NOVA intends to allow network services and functions, by a variety of developers, to be published and brokered/traded, allowing customers to browse and select services that best match their needs (negotiating SLAs and billing models). One of the main marketplace functions is the publication of resources and network function advertisements in order to compose and package NFV services. VBaaS can enhance VNFs capabilities publication and certification models of their performance when constructing a diverse set of VNF-Profiles. Consequently, VNF flavors can be defined by an enhanced set of VBaaS capabilities for smarter placements, e.g., exposing VBaaS interfaces for developers evolve VNF-Profiles.

Unify [Szab15] presents a recursive compute and network joint virtualization and programming view of generic infrastructures (e.g., data centers), considering NFV as an example. Big Switch with Big Software (BiS-BiS) represents the abstraction of such virtualization. The draft presents an Yang model to describe a Network Function Forwarding Graph (NF-FG) representing NFV resources. Complementary, as part of main Unify goals, problem statements and challenges about the unification of carrier and cloud networks [Csas15] show the need for a high level of programmability beyond policy and service descriptions. As presented, VBaaS can contribute for such abstractions since it detaches from monolithic views of infrastructure resources allowing the recursive definitions of VBaaS Service Graphs and, consequently, the decomposition of benchmark tasks in multiple domains. Besides, as stated by Unify measurements

and analytics challenges, VBaaS poses as a solution of tools for planning, testing, and reliability assurance of NFVIs.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBD

10. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil.

This work is partially supported by FP7 UNIFY, a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

11. Informative References

[BVLS15] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations", Communications Magazine, IEEE, vol. 53, no. 2, pp. 90-97 , February 2015.

[Csas15] R. Szabo, A. Csaszar, K. Pentikousis, M. Kind, D. Daino, Z. Qiang, H. Woesner, "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", March 2015, <https://tools.ietf.org/html/draft-unify-nfvrg-challenges-01>.

[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, "Management and Orchestration - ETSI GS NFV-MAN 001 V1.1.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf>.

[ETS14d] ETSI, "Virtual Network Functions Architecture - ETSI GS NFV-SWA 001 V1.1.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV-SWA001v010101p.pdf>.

[ETS14t] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V0.0.13", Sept 2015, <http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0013.zip>.

[Meir15] C. Meirosu, A. Manzalini, J. Kim, R. Steinert, S. Sharma, G. Marchetto, I. Papafili, K. Pentikousis, S. Wright, "DevOps for Software-Defined Telecom Infrastructures", July 2015, <http://www.ietf.org/id/draft-unify-nfvrg-devops-02.txt>.

[Mor15] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", February 2015, <http://tools.ietf.org/html/draft-morton-bmwg-virtual-net-03>.

[PSPL15] E. Paik and M-K. Shin and S. Pack and S. Lee, "Resource Management in Service Chaining", March 2015, <http://tools.ietf.org/html/draft-lee-nfvrg-resource-management-service-chain-01>.

[Szab15] R. Szabo, Z. Qiang, M. Kind, "Towards recursive virtualization and programming for network and cloud resources", July 2015, <https://tools.ietf.org/html/draft-unify-nfvrg-recursive-programming-01>.

[Tah15] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in OPNFV", July 2015, <https://tools.ietf.org/html/draft-vsperf-bmwg-vswitch-opnfv-00>.

[TNOVA15] T-NOVA, "T-NOVA Results", July 2015, <http://www.t-nova.eu/results/>.

[Yard15] OPNFV, "Project: Yardstick - Infrastructure Verification", July 2015, <https://wiki.opnfv.org/yardstick>.

Authors' Addresses

Raphael Vicente Rosa
University of Campinas
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: raphaelvrosa@dca.fee.unicamp.br
URI: http://www.intrig.dca.fee.unicamp.br/

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas 13083-852
Brasil

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

Robert Szabo
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: robert.szabo@ericsson.com
URI: http://www.ericsson.com/