Abstract

The introduction of network and service functionality virtualization in carrier-grade networks promises improved operations in terms of flexibility, efficiency, and manageability. In current practice, virtualization is controlled through orchestrator entities that expose programmable interfaces according to the underlying resource types. Typically this means the adoption of, on the one hand, established data center compute/storage and, on the other, network control APIs which were originally developed in isolation. Arguably, the possibility for innovation highly depends on the capabilities and openness of the aforementioned interfaces. This document introduces in simple terms the problems arising when one follows this approach and motivates the need for a high level of programmability beyond policy and service descriptions. This document also summarizes the challenges related to orchestration programming in this unified cloud and carrier network production environment.

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/.
1. Introduction

To a large degree there is agreement in the network research, practitioner, and standardization communities that rigid network control limits the flexibility and manageability of speedy service creation, as discussed in [NSC] and the references therein. For instance, it is not unusual that today an average service creation time cycle exceeds 90 hours, whereas given the recent advances in
virtualization and cloudification one would be interested in service creation times in the order of minutes [EU-5GPPP-Contract] if not seconds.

Flexible service definition and creation start by formalizing the service into the concept of network function forwarding graphs, such as the ETSI VNF Forwarding Graph [ETSI-NFV-Arch] or the ongoing work in IETF [I-D.ietf-sfc-problem-statement]. These graphs represent the way in which service end-points (e.g., customer access) are interconnected with a set of selected network functionalities such as firewalls, load balancers, and so on, to deliver a network service. Service graph representations form the input for the management and orchestration to instantiate and configure the requested service. For example, ETSI defined a Management and Orchestration (MANO) framework in [ETSI-NFV-MANO]. We note that throughout such a management and orchestration framework different abstractions may appear for separation of concerns, roles or functionality, or for information hiding.

Compute virtualization is central to the concept of Network Function Virtualization (NFV). However, carrier-grade services demand that all components of the data path, such as Network Functions (NFs), virtual NFs (VNFs) and virtual links, meet key performance requirements. In this context, the inclusion of Data Center (DC) platforms, such as OpenStack [OpenStack], into the SDN infrastructure is far from trivial.

In this document we examine the problems arising as one combines these two formerly isolated environments in an effort to create a unified production environment and discuss the associated emerging challenges. Our goal is the definition of a production environment that allows multi-vendor and multi-domain operation based on open and interoperable implementations of the key entities described in the remainder of this document.

2. Terms and Definitions

We use the term compute and "compute and storage" interchangeably throughout the document. Moreover, we use the following definitions, as established in [ETSI-NFV-Arch]:

**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.

VNF: Virtualized Network Function - a software-based network function.

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 [ETSI-NFV-MANO], this is the global entity responsible for management and orchestration of NFV lifecycle.

Further, we make use of the following terms:

NF: a network function, either software-based (VNF) or appliance-based.

SW: a (routing/switching) network element with a programmable control plane interface.

DC: a data center network element which in addition to a programmable control plane interface offers a DC control interface.

LSI: Logical Switch Instance - a software switch instance.

3. Motivations

Figure 1 illustrates a simple service graph comprising three network functions (NFs). For the sake of simplicity, we will assume only two types of infrastructure resources, namely SWs and DCs as per the terminology introduced above, and ignore appliance-based NFs for the time being. The goal is to implement the given service based on the available infrastructure resources.

```
fr2  ----+ fr3
      +->|NF2|---o--+
           |  4  ----+ 5 |
           +----+   V +----+
      1-->|NF1|---o--------o-|NF3|---o--8
           2 ----+ 3     fr1  6 ----+ 7
```

Figure 1: Service graph

The service graph definition contains NF types (NF1, NF2, NF3) along with the
- corresponding ports (NF1:{2,3}; NF2:{4,5}; NF3:{6,7})
- service access points {1,8} corresponding to infrastructure resources,
- definition of forwarding behavior (fr1, fr2, fr3)

The forwarding behavior contains classifications for matching of traffic flows and corresponding outbound forwarding actions.

Assume now that we would like to use the infrastructure (topology, network and software resources) depicted in Figure 2 and Figure 3 to implement the aforementioned service graph. That is, we have three SWs and two Points of Presence (PoPs) with DC software resources at our disposal.

![Figure 2: Infrastructure resources](image)

![Figure 3: A virtualized Point of Presence (PoP) with software resources (Compute Node - CN)](image)

In the simplest case, all resources would be part of the same service provider (SP) domain. We need to ensure that each entity in Figure 2 can be procured from a different vendor and therefore interoperability is key for multi-vendor NFVI deployment. Without
such interoperability different technologies for data center and network operation result in distinct technology domains within a single carrier. Multi-technology barriers start to emerge hindering the full programmability of the NFVI and limiting the potential for rapid service deployment.

We are also interested in a multi-operation environment, where the roles and responsibilities are distributed according to some organizational structure within the organization. Finally, we are interested in multi-provider environment, where different infrastructure resources are available from different service providers (SPs). Figure 2 indicates a multi-provider environment in the lower part of the figure as an example. We expect that this type of deployments will become more common in the future as they are well suited with the elasticity and flexibility requirements [NSC].

Figure 2 also shows the service access points corresponding to the overarching domain view, i.e., $\{1,8\}$. In order to deploy the service graph of Figure 1 on the infrastructure resources of Figure 2, we will need an appropriate mapping which can be implemented in practice. In Figure 4 we illustrate a resource orchestrator (RO) as a functional entity whose task is to map the service graph to the infrastructure resources under some service constraints and taking into account the NF resource descriptions.
NF resource descriptions are assumed to contain information necessary to map NF types to a choice of instantiable VNF flavor or a selection of an already deployed NF appliance and networking demands for different operational policies. For example, if energy efficiency is to be considered during the decision process then information related to energy consumption of different NF flavors under different conditions (e.g., network load) should be included in the resource description.

Note that we also introduce a new service provider (SP0) which effectively operates on top of the virtualized infrastructure offered by SP1, SP2 and SP3.

In order for the RO to execute the resource mapping (which in general is a hard problem) it needs to operate on the combined control plane illustrated in Figure 5. In this figure we mark clearly that the interfaces to the compute (DC) control plane and the SDN (SW) control
plane are distinct and implemented through different interfaces/APIs. For example, Ic1 could be the Apache CloudStack API, while Ic2 could be a control plane protocol such as ForCES or OpenFlow [I-D.irtf-sdnrg-layer-terminology]. In this case, the orchestrator at SP0 (top part of the figure) needs to maintain a tight coordination across this range of interfaces.

```
+---------+
|Orchestr.|
| SP0 |
+---------+

Ic1 V |SDN Ctrl| V Ic3

--------+  SP2  +--------
|Comp Ctrl|  +----+  |Comp Ctrl|
| SP1 |

--------+  V  +----+

V |

--------+  V |

1 |PoP |  +----+  +----+  |PoP | 8
0--|DC1 |  -----|SW2|  -----|SW4|  ----|DC2|--o

[----SP1----][--------SP2--------][---SP3----]
[------------------------SP0------------------]
```

Figure 5: The RO Control Plane view. Control plane interfaces are indicated with (line) arrows. Data plane connections are indicated with simple lines.

In the real-world, however, orchestration operations do not stop, for example, at the DC1 level as depicted in Figure 5. If we (so-to-speak) "zoom into" DC1 we will see a similar pattern and the need to coordinate SW and DC resources within DC1 as illustrated in Figure 6. As depicted, this edge PoP includes compute nodes (CNs) and SWs which in most of the cases will also contain an internal topology.

In Figure 6, IcA is an interface similar to Ic2 in Figure 5, while IcB could be, for example, OpenStack Nova or similar. The Northbound Interface (NBI) to the Compute Controller can use Ic1 or Ic3 as shown in Figure 5.
In turn, each single Compute Node (CN) may also have internal switching resources (see Figure 7). In a carrier environment, in order to meet data path requirements, allocation of compute node internal distributed resources (blades, CPU cores, etc.) may become equivalently important.

Figure 6: PoP DC Network with Compute Nodes (CN)

Figure 7: Compute Node with internal switching resource
4. Problem Statement

The motivational examples of Section 3 illustrate that compute virtualization implicitly involves network virtualization. On the other hand, if one starts with an SDN network and adds compute resources to network elements, then compute resources must be assigned to some virtualized network resources if offered to clients. That is, we observe that compute virtualization is implicitly associated with network virtualization. Furthermore, virtualization leads to recursions with clients (redefining and) reselling resources and services [I-D.huang-sfc-use-case-recursive-service].

We argue that given the multi-level virtualization of compute, storage and network domains, automation of the corresponding resource provisioning needs a recursive programmatic interface. The current separated compute and network programming interfaces cannot provide such recursions and cannot satisfy key requirements for multi-vendor, multi-technology and multi-provider interoperability environments. Therefore we foresee the necessity of a recursive programmatic interface for joint compute, storage and network provisioning.

5. Challenges

We summarize in this section the key questions and challenges, which we hope will initiate further discussions in the NFVRG community.

5.1. Orchestration

Firstly, as motivated in Section 3, orchestrating networking resources appears to have a recursive nature at different levels of the hierarchy. Would a programmatic interface at the combined compute and network abstraction better support this recursive and constraint-based resource allocation?

Secondly, can such a joint compute, storage and network programmatic interface allow an automated resource orchestration similar to a recursive SDN architecture [ONF-SDN-ARCH]?

5.2. Resource description

Prerequisite for joint placement decisions of compute, storage and network is the adequate description of available resources. This means that the interfaces (IcA, IcB etc. in Figure 5 and Figure 6) are of bidirectional nature, exposing resources as well as reserving. There have been manifold attempts to create frameworks for resource description, most prominently RDF of W3C, NDL, the GENI RPC and its concept of Aggregate Managers, ONF’s TTP and many more.
Quite naturally, all attempts to standardize "arbitrary" resource descriptions lead to creating ontologies, complex graphs describing relations of terms to each other.

Practical descriptions of compute resources are currently focusing on number of logical CPU cores, available RAM and storage, allowing, e.g., the OpenStack Nova scheduler to meet placement decisions. In heterogeneous network and compute environments, hardware may have different acceleration capabilities (e.g., AES-NI or hardware random number generators), so the notion of logical compute cores is not expressive enough. In addition, the network interfaces (and link load) provide important information on how fast a certain VNF can be executed in one node.

This may lead to a description of resources as VNF-FGs themselves. Networking resource (SW) may expose the capability to forward and process frames in, e.g., OpenFlow TableFeatures reply. Compute nodes in the VNF-FG would expose lists of capabilities like the presence of AES hardware acceleration, Intel DPDK support, or complex functions like a running web server. An essential part of the compute node’s capability would be the ability to run a certain VNF of type X within a certain QoS spec. As the QoS is service specific, it can only be exposed by a control function within the instantiated VNF-FG.

5.3. Dependencies (de-composition)

Salt [SALT], Puppet [PUPPET], Chef [CHEF] and Ansible [ANSIBLE] are tools to manage large scale installations of virtual machines in DC environments. Essentially, the decomposition of a complex function into its dependencies is encoded in "recipes" (Chef).

OASIS TOSCA [TOSCA] specification aims at describing application layer services to automate interoperable deployment in alternative cloud environments. The TOSCA specification "provides a language to describe service components and their relationships using a service topology".

Is there a dependency (decomposition) abstraction suitable to drive resource orchestration between application layer descriptions (like TOSCA) and cloud specific installations (like Chef recipes)?

5.4. Elastic VNF

In many use cases, a VNF may not be designed for scaling up/down, as scaling up/down may require a restart of the VNF which the state data may be lost. Normally a VNF may be capable for scaling in/out only. Such VNF is designed running on top of a small VM and grouped as a pool of one VNF function. VNF scaling may crossing multiple NFVI
PoPs (or data center)s in order to avoid limitation of the NVFI capability. At cross DC scaling, the result is that the new VNF instance may be placed at a remote cloud location. At VNF scaling, it is a must requirement to provide the same level of Service Level Agreement (SLA) including performance, reliability and security.

In general, a VNF is part of a VNF Forwarding Graph (VNF FG), meaning the data traffic may traverse multiple stateful and stateless VNF functions in sequence. When some VNF instances of a given service function chain are placed / scaled out in a distant cloud execution, the service traffic may have to traverse multiple VNF instances which are located in multiple physical locations. In the worst case, the data traffic may ping-pong between multiple physical locations. Therefore it is important to take the whole service function chain’s performance into consideration when placing and scaling one of its VNF instance. Network and cloud resources need mutual considerations, see [I-D.zu-nfvrg-elasticity-vnf].

5.5. Measurement and analytics

Programmable, dynamic, and elastic VNF deployment requires that the Resource Orchestrator (RO) entities obtain timely information about the actual operational conditions between different locations where VNFs can be placed. Scaling VNFs in/out/up/down, VNF execution migration and VNF mobility, as well as right-sizing the VNFI resource allocations is a research area that is expected to grow in the coming years as mechanisms, heuristics, and measurement and analytics frameworks are developed.

For example, Veitch et al. [IAF] point out that NFV deployment will "present network operators with significant implementation challenges". They look into the problems arising from the lack of proper tools for testing and diagnostics and explore the use of embedded instrumentation. They find that in certain scenarios fine-tuning resource allocation based on instrumentation can lead to at least 50% reduction in compute provisioning. In this context, three categories emerge where more research is needed.

First, in the compute domain, performance analysis will need to evolve significantly from the current "safety factor" mentality which has served well carriers in the dedicated, hardware-based appliances era. In the emerging softwarized deployments, VNFI will require new tools for planning, testing, and reliability assurance.

Second, in the network domain, performance measurement and analysis will play a key role in determining the scope and range of VNF distribution across the resources available. For example, IETF has worked on the standardization of IP performance metrics for years.
The Two-Way Active Measurement Protocol (TWAMP) could be employed, for instance, to capture the actual operational state of the network prior to making RO decisions. TWAMP management, however, still lacks a standardized and programmable management and configuration data model. We expect that as VNFI programmability gathers interest from network carriers several IETF protocols will be revisited in order to bring them up to date with respect to the current operational requirements. To this end, NFVRG can play an active role in identifying future IETF standardization directions.

Third, non-technical considerations which relate to business aspects or priorities need to be modeled and codified so that ROs can take intelligent decisions. Energy efficiency and cost, for example, can steer NFV placement. In NFVI deployments operational practices such as follow-the-sun will be considered as earlier research in the data center context implies.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

TBD

8. Acknowledgement

The authors would like to thank the UNIFY team for inspiring discussions and in particular Fritz-Joachim Westphal for his comments and suggestions on how to refine this draft.

This work is 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.

9. Informative References


[ETSI-NFV-Arch]
ETSI, "Architectural Framework v1.1.1", Oct 2013,
<http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf>.

[ETSI-NFV-MANO]
ETSI, "Network Function Virtualization (NFV) Management and Orchestration V0.6.1 (draft)", Jul. 2014,

[EU-5GPPP-Contract]
5G-PPP Association, "Contractual Arrangement: Setting up a Public-Private Partnership in the Area of Advance 5G Network Infrastructure for the Future Internet between the European Union and the 5G Infrastructure Association", Dec 2013,
<http://5g-ppp.eu/contract/>.

[I-D.huang-sfc-use-case-recursive-service]

[I-D.iertf-sfc-problem-statement]

[I-D.irtf-sdnrg-layer-terminology]

[I-D.zu-nfvrg-elasticity-vnf]
Qiang, Z. and R. Szabo, "Elasticity VNF", draft-zu-nfvrg-elasticity-vnf-01 (work in progress), March 2015.


[ONF-SDN-ARCH]
ONF, "SDN architecture", Jun. 2014,

[OpenStack]
The OpenStack project, "Openstack cloud software", 2014,


[SALT] SaltStack, "Salt (Documentation)", 2015,

<http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/ TOSCA-v1.0-os.html>.

Authors’ Addresses

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

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

Andras Csaszar
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: andras.csaszar@ericsson.com
URI: http://www.ericsson.com/