NFVRG
Internet-Draft
Intended status: Informational
Expires: September 10, 2017

P. Aranda Gutierrez
UC3M
D. Lopez
Telefonica
S. Salsano
Univ. of Rome Tor Vergata/CNIT
E. Batanero
Telefonica

March 09, 2017

High-level VNF Descriptors using NEMO
draft-aranda-nfvrg-recursive-vnf-03

Abstract

Current efforts in the scope of Network Function Virtualisation(NFV) propose YAML-based descriptors for Virtual Network Functions (VNFs) and for their composition in Network Services (NS) These descriptors are human-readable but hardly understandable by humans. On the other hand, there has been an effort proposed to the IETF to define a human-readable (and understandable) representation for networks, known as NEMO. In this draft, we propose a simple extension to NEMO to accommodate VNF Descriptors (VNFDs) in a similar manner as inline assembly is integrated in higher-level programming languages.

This approach enables the creation of recursive VNF forwarding graphs in Service Descriptors, practically making them recursive.

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 September 10, 2017.

Table of Contents

1. Introduction

Currently, there is a lot of on-going activity to deploy NFV in the network. From the point of view of the orchestration, Virtual Network Functions are blocks that are deployed in the infrastructure as independent units. Following the reference architectural model proposed in [ETSI-NFV-MANO], VNFs provide for one layer of components (VNF components(VNFCs)) below, i.e. a set of VNFCs accessible to a VNF provider can be composed into VNFs. However, there is no simple

way to use existing VNFs as components in VNFs with a higher degree of complexity. In addition, Network Service Descriptors (NSD) and VNF Descriptors (VNFDs) specified in [ETSI-NFV-MANO] and used in different open source MANO frameworks are YAML-based files, which despite being human readable, are not easy to understand.

On the other hand, there has been recently an attempt to work on a modelling language for networks or Network Modelling (NEMO) language. This language is human-readable and provides constructs that support recursiveness. In this draft, we propose an addition to NEMO to make it interact with VNFDs supported by a NFV MANO framework. This integration creates a new language for VNFDs that is recursive, allowing VNFs to be created based on the definitions of existing VNFs.

This draft uses two example formats to show how low level descriptors can be imported into NEMO. The first one is the format used in the OpenMANO [1] framework. The second one follows strictly the specifications provided by ETSI NFV ISG in [ETSI-NFV-MANO]. Conceptually, other descriptor formats like TOSCA can also be used at this level.

2. Terminology and abbreviations

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Prior art

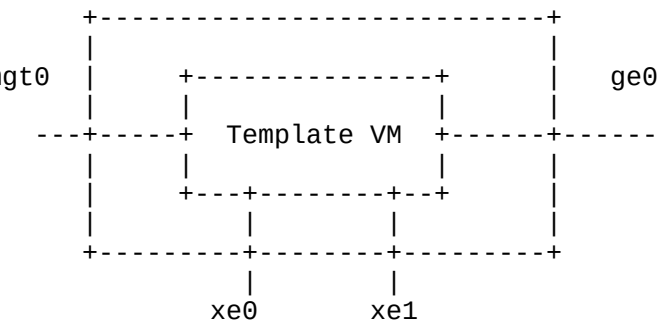
3.1. Virtual network function descriptors

Virtual network function descriptors (VNFDs) are used in the Management and orchestration (MANO) framework of the ETSI NFV to achieve the optimal deployment of virtual network functions (VNFs). The Virtual Infrastructure Manager (VIM) uses this information to place the functions optimally. VNFDs include information of the components of a specific VNF and their interconnection to implement the VNF, in the form of a forwarding graph. In addition to the forwarding graph, the VNFD includes information regarding the interfaces of the VNF. These are then used to connect the VNF to either physical or logical interfaces once it is deployed.

There are different MANO frameworks available. For this draft, we will first concentrate on the example of OpenMANO [2], which uses a YAML [3] representation similar to the one specified in [ETSI-NFV-MANO]. Then we will provide an example using the exact format specified in [ETSI-NFV-MANO].

3.1.1. OpenMANO VNFDs

Taking the example from the (public) OpenMANO github repository, we can easily identify the virtual interfaces of the sample VNFs in their descriptors:




```
vnf:
    name: TEMPLATE
    description: This is a template to help in the creation of
    # class: parent      # Optional. Used to organize VNFs
    external-connections:
    -   name:              mgmt0
        type:              mgmt
        VNFC:              TEMPLATE-VM
        local_iface_name:  mgmt0
        description:       Management interface
    -   name:              xe0
        type:              data
        VNFC:              TEMPLATE-VM
        local_iface_name:  xe0
        description:       Data interface 1
    -   name:              xe1
        type:              data
        VNFC:              TEMPLATE-VM
        local_iface_name:  xe1
        description:       Data interface 2
    -   name:              ge0
        type:              bridge
        VNFC:              TEMPLATE-VM
        local_iface_name:  ge0
        description:       Bridge interface
```

Figure 1: Sample VNF and descriptor (source: OpenMANO github)

3.1.2. ETSI MANO VNFDs

In this example we consider the VNF represented in Figure 6.4 of [ETSI-NFV-MANO]. Its internal diagram, including a VNF component, is represented in Figure Figure 2. A YAML representation of the VNF Descriptor is reported in Figure Figure 3. The topology of the interconnection of VNFs is expressed by using the abstraction of Virtual Links, which interconnect Connection Points of the VNFs. The Virtual Links are described by Virtual Link Descriptors (VLD) files. An example YAML representation of the Virtual Link VL1 in the example VNF is reported in Figure Figure 3. In order to understand the topology, a (potentially large) set of VNFD and VLD files needs to be analysed. For a human programmer of the service, this representation is not friendly to write and very hard to read/understand/debug.

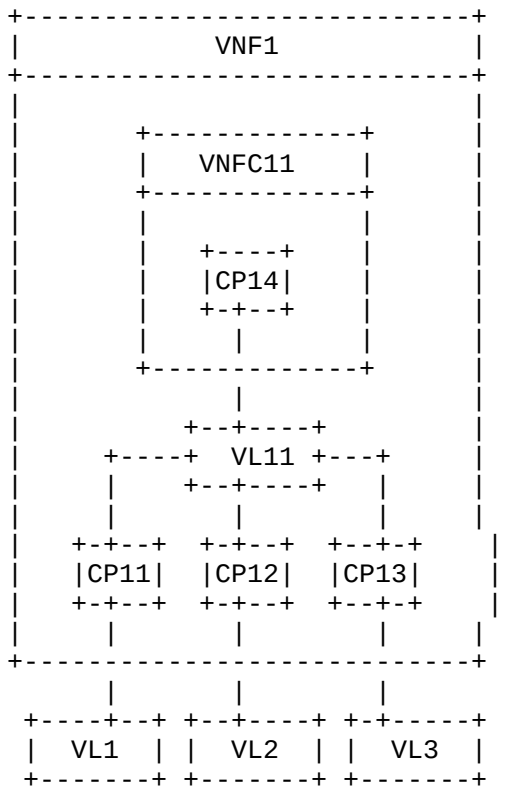


Figure 2: VNF example

```
#######################################
# VNF Descriptor of a VNF called vnf1
#######################################
id: vnf1
description_version: '0.1'
vendor: netgroup
version: '0.1'
connection_point:
- id: cp11
  type: ''
  virtual_link_reference: vl11
- id: cp12
  type: ''
  virtual_link_reference: vl11
- id: cp13
  type: ''
  virtual_link_reference: vl11
vdu:
- id: vdu11
  computation_requirement: ''
  virtual_memory_resource_element: ''
  virtual_network_bandwidth_resource: ''
  vnfc:
  - id: vnfc11
    connection_point:
    - id: cp14
      type: NIC
      virtual_link_reference: vl11
virtual_link:
- id: vl11
  connection_points_references:
  - cp11
  - cp12
  - cp13
  - cp14
  connectivity_type: ' E-Line'
  root_requirement: ''
```

Figure 3: ETSI MANO compliant VNF descriptor example

```
##########################################
# Virtual Link Descriptor of a VL called vl1
##########################################
id: vl1
descriptor_version: '0.1'
test_access: none
vendor: netgroup
connection:
- cp01
- cp11
connectivity_type: E-LAN
number_of_endpoints: 2
root_requirement: ''
```

Figure 4: ETSI MANO compliant Virtual Link descriptor example

3.2. NEMO

The Network Modeling (NEMO) language is described in [I-D.xia-sdnrg-nemo-language]. It provides a simple way of describing network scenarios. The language is based on a two-stage process. In the first stage, models for nodes, links and other entities are defined. In the second stage, the defined models are instantiated. The NEMO language also allows for behavioural descriptions. A variant of the NEMO language is used in the OpenDaylight NEMO northbound API [4].

NEMO allows to define NodeModels, which are then instantiated in the infrastructure. NodeModels are recursive and can be build with basic node types or with previously defined NodeModels. An example for a script defining a NodeModel is shown below:

```
CREATE NodeModel dmz
  Property string: location-fw, string: location-n2,
    string: ipprefix, string: gatewayip, string: srcip,
    string: subnodes-n2;
  Node fw1
    Type fw
    Property location: location-fw,
      operating-mode: layer3;
...
```

Figure 5: Creating a NodeModel in NEMO

4. Additional requirements on NEMO

In order to integrate VNFDs into NEMO, we need to take into account two specifics of VNFDs, which cannot be expressed in the current language model. Firstly, we need a way to reference the file which holds the VNFD provided by the VNF developer. This will normally be a universal resource identifier (URI). Additionally, we need to make the NEMO model aware of the virtual network interfaces.

4.1. Referencing VNFDs in a NodeModel

As explained in the introduction, in order integrate VNFDs into the NEMO language in the easiest way we need to reference the VNFD as a Universal Resource Identifier (URI) as defined in RFC 3986 [RFC3986]. To this avail, we define a new element in the NodeModel to import the VNFD:

```
CREATE NodeModel <node_model_name> VNFD <vnfd_uri>;
```

4.2. Referencing the network interfaces of a VNF in a NodeModel

As shown in Figure 1, VNFDs include an exhaustive list of interfaces, including the interfaces to the management network. However, since these interfaces may not be significant for specific network scenarios and since interface names in the VNFD may not be adequate in NEMO, we propose to define a new entity, namely the ConnectionPoint, which is included in the node model .

```
CREATE NodeModel <node_model_name>;
  ConnectionPoint <cp_name> at VNFD:<iface_from_vnfd>;
```

4.3. An example

Once these two elements are included in the NEMO language, it is possibly to recursively define NodeModel elements that use VNFDs in the lowest level of recursion. Firstly, we create NodeModels from VNFDs:

```
CREATE NodeModel sample_vnf VNFD https://github.com/nfvlabs
/openmano.git/openmano/vnfs/examples/dataplaneVNF1.yaml;
    ConnectionPoint data_inside at VNFD:ge0;
    ConnectionPoint data_outside at VNFD:ge1;
```

Import from a sample VNFD from the OpenMANO repository

Then we can reuse these NodeModels recursively to create complex NodeModels:

```
CREATE NodeModel complex_vnf;
    Node input_vnf Type sample_vnf;
    Node output_vnf Type shaper_vnf;
    ConnectionPoint input;
    ConnectionPoint output
    Connection icon Type p2p Endnodes input, input_vnf:data_inside;
    Connection ocon Type p2p Endnodes output, output_vnf:wan;
    Connection intn Type p2p input_vnf:data_outside, output_vnf:lan;
```

Create a composed NodeModel

This NodeModel definition creates a composed model linking the sample_vnf created from the VNFD with a hypothetical shaper_vnf defined elsewhere. This definition can be represented graphically as follows:

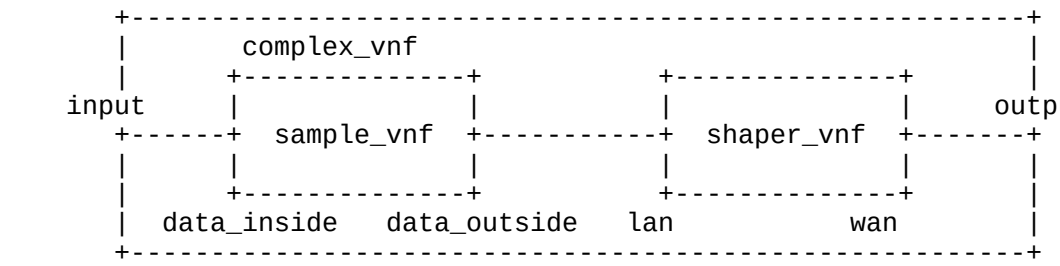


Figure 6

In ETSI NFV, a network service is described by one or more VNFs that are connected through one or more network VNFFGs. This is no more than what is defined in the composed NodeModel shown if Figure 6. By using NEMO, we provide a simple way to define VNF forwarding graphs (VNF-FGs) in network service descriptors in a recursive way.

5. Conclusion

With the strategy defined in this document, we are able to link a low-level VNF description into a high-level description language for networks like NEMO. Effectively, we are introducing recursiveness in VNFDs, allowing complex service descriptors to be built by reusing previously tested descriptors graphs as building blocks.

Although we have used the OpenMANO descriptor format in this document, other descriptors and concepts (i.e. as those used by TOSCA

[5]) can also be used as the lowest level in this extension to the NEMO language.

6. IANA Considerations

This draft includes no request to IANA.

7. Security Considerations

The VNFD construct as IMPORT allows referencing external resources. Developers using it in NEMO scripts are advised to verify the source of those external resources, and whenever possible, rely on sources with a verifiable identity through cryptographic methods.

8. Acknowledgement

This work has been partially performed in the scope of the SUPERFLUIDITY project, which has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No.671566 (Research and Innovation Action).

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[ETSI-NFV-MANO] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI GS NFV-MAN 001 V1.1.1 (2014-12), December 2014.

9.2. Informative References

[I-D.xia-sdnrg-nemo-language] Xia, Y., Jiang, S., Zhou, T., Hares, S., and Y. Zhang, "NEMO (NEtwork MOdeling) Language", draft-xia-sdnrg-nemo-language-04 (work in progress), April 2016.

9.3. URIs

[1] https://github.com/nfvlabs/openmano

[2] https://github.com/nfvlabs/openmano

[3] yaml.org

[4] https://wiki.opendaylight.org/view/NEMO:Main

[5] http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html

Authors' Addresses

Pedro A. Aranda Gutierrez
Universidad Carlos III Madrid
Leganes 28911
Spain

Email: paranda@it.uc3m.es

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid 28010
Spain

Email: diego.r.lopez@telefonica.com

Stefano Salsano
Univ. of Rome Tor Vergata/CNIT
Via del Politecnico, 1
Rome 00133
Italy

Email: stefano.salsano@uniroma2.it

Elena Batanero
Telefonica I+D
Zurbaran, 12
Madrid 28010
Spain

Email: elena.batanerogarcia@telefonica.com

NFV RG
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

CJ. Bernardos
UC3M
LM. Contreras
TID
I. Vaishnavi
Huawei
R. Szabo
Ericsson

March 13, 2017

Multi-domain Network Virtualization
draft-bernardos-nfvrg-multidomain-02

Abstract

This draft analyzes the problem of multi-provider multi-domain orchestration, by first scoping the problem, then looking into potential architectural approaches, and finally describing the solutions being developed by the European 5GEx project.

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 September 14, 2017.

Table of Contents

1. Introduction

The telecommunications sector is experiencing a major revolution that will shape the way networks and services are designed and deployed for the next decade. We are witnessing an explosion in the number of applications and services demanded by users, which are now really capable of accessing them on the move. In order to cope with such a demand, some network operators are looking at the cloud computing paradigm, which enables a potential reduction of the overall costs by outsourcing communication services from specific hardware in the operator's core to server farms scattered in datacenters. These services have different characteristics if compared with conventional IT services that have to be taken into account in this cloudification process. Also the transport network is affected in that it is evolving to a more sophisticated form of IP architecture with trends like separation of control and data plane traffic, and more fine-grained forwarding of packets (beyond looking at the destination IP address) in the network to fulfill new business and service goals.

Virtualization of functions also provides operators with tools to deploy new services much faster, as compared to the traditional use of monolithic and tightly integrated dedicated machinery. As a natural next step, mobile network operators need to re-think how to

evolve their existing network infrastructures and how to deploy new ones to address the challenges posed by the increasing customers' demands, as well as by the huge competition among operators. All these changes are triggering the need for a modification in the way operators and infrastructure providers operate their networks, as they need to significantly reduce the costs incurred in deploying a new service and operating it. Some of the mechanisms that are being considered and already adopted by operators include: sharing of network infrastructure to reduce costs, virtualization of core servers running in data centers as a way of supporting their load-aware elastic dimensioning, and dynamic energy policies to reduce the monthly electricity bill. However, this has proved to be tough to put in practice, and not enough. Indeed, it is not easy to deploy new mechanisms in a running operational network due to the high dependency on proprietary (and sometime obscure) protocols and interfaces, which are complex to manage and often require configuring multiple devices in a decentralized way.

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are changing the way the telecommunications sector will deploy, extend and operate their networks. TBD: add multi-domain.

2. Terminology

The following terms used in this document are defined by the ETSI NVF ISG, and the ONF and the IETF:

NFV Infrastructure (NFVI): totality of all hardware and software components which build up the environment in which VNFs are deployed

NFV Management and Orchestration (NFV-MANO): functions collectively provided by NFVO, VNFM, and VIM.

NFV Orchestrator (NFVO): 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.

Network Service Orchestration (NSO): function responsible for the Network Service lifecycle management, including operations such as: On-board Network Service, Instantiate Network Service, Scale Network Service, Update Network Service, etc.

OpenFlow protocol (OFP): allowing vendor independent programming of control functions in network nodes.

Resource Orchestration (RO): subset of NFV Orchestrator functions that are responsible for global resource management governance.

Service Function Chain (SFC): for a given service, the abstracted view of the required service functions and the order in which they are to be applied. This is somehow equivalent to the Network Function Forwarding Graph (NF-FG) at ETSI.

Service Function Path (SFP): the selection of specific service function instances on specific network nodes to form a service graph through which an SFC is instantiated.

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

Virtualized Network Function (VNF): implementation of a Network Function that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

Virtualized Network Function Manager (VNFM): functional block that is responsible for the lifecycle management of VNF.

3. Background: the ETSI NFV architecture

The ETSI ISG NFV is a working group which, since 2012, aims to evolve quasi-standard IT virtualization technology to consolidate many network equipment types into industry standard high volume servers, switches, and storage. It enables implementing network functions in software that can run on a range of industry standard server hardware and can be moved to, or loaded in, various locations in the network as required, without the need to install new equipment. To date, ETSI NFV is by far the most accepted NFV reference framework and architectural footprint [etsi_nvf_whitepaper]. The ETSI NFV framework architecture framework is composed of three domains (Figure 1):

- Virtualized Network Function, running over the NFVI.
- NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualized. NFVI supports the execution of the VNFs.
- NFV Management and Orchestration, which covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs. NFV Management and Orchestration focuses on

all virtualization specific management tasks necessary in the NFV framework.

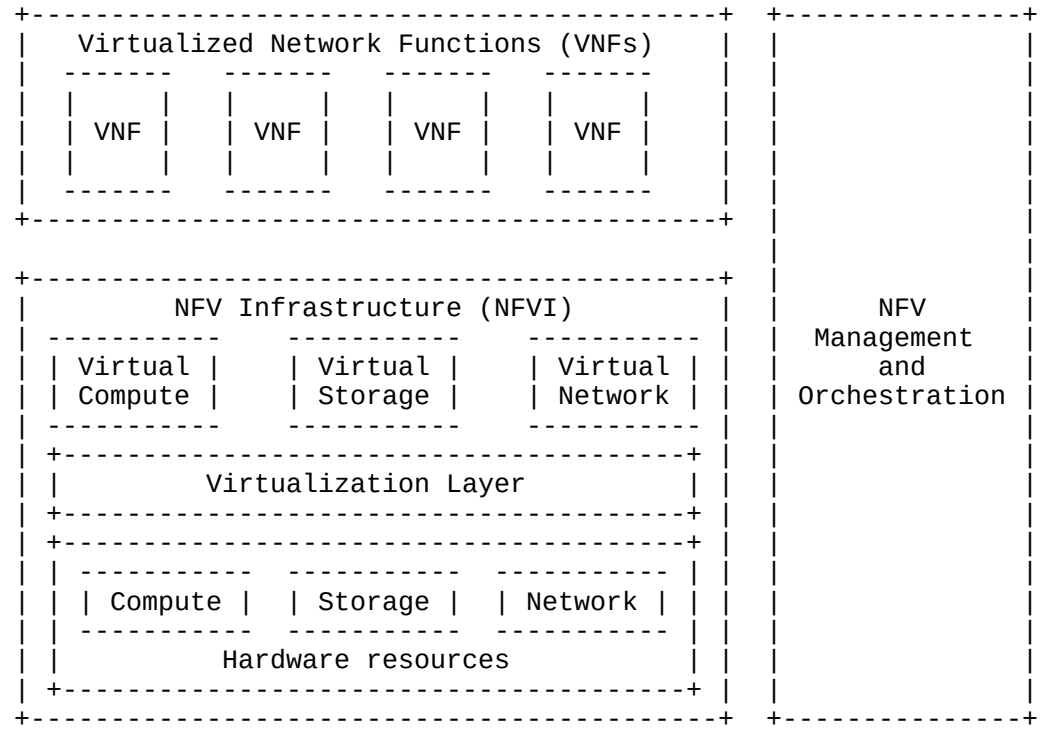


Figure 1: ETSI NFV framework

The NFV architectural framework identifies functional blocks and the main reference points between such blocks. Some of these are already present in current deployments, whilst others might be necessary additions in order to support the virtualization process and consequent operation. The functional blocks are (Figure 2):

- Virtualized Network Function (VNF).
- Element Management (EM).
- NFV Infrastructure, including: Hardware and virtualized resources, and Virtualization Layer.
- Virtualized Infrastructure Manager(s) (VIM).
- NFV Orchestrator.

- o VNF Manager(s).
- o Service, VNF and Infrastructure Description.
- o Operations and Business Support Systems (OSS/BSS).

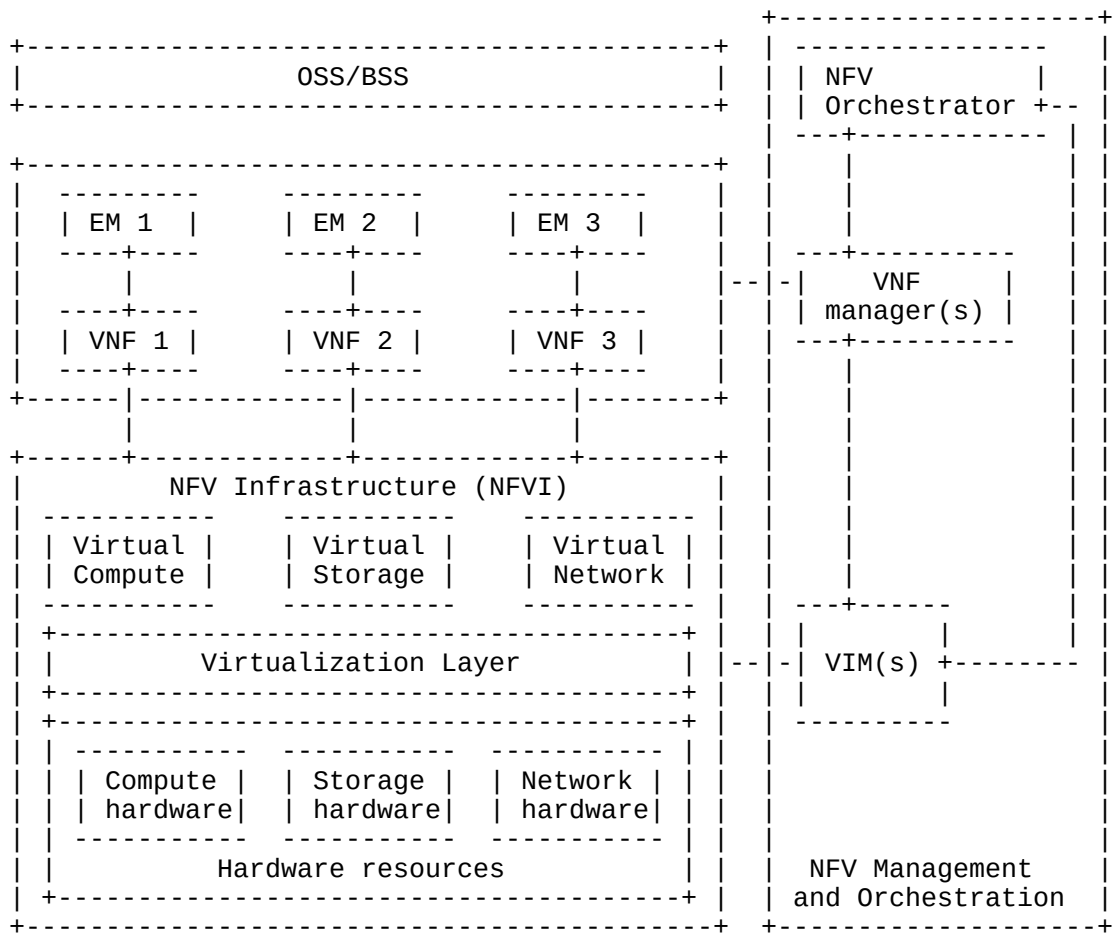


Figure 2: ETSI NFV reference architecture

4. Multidomain problem statement

Market fragmentation results from having a multitude of telecommunications network and cloud operators each with a footprint focused to a specific region. This makes it difficult to deploy cost effective infrastructure services, such as virtual connectivity or compute resources, spanning multiple countries as no single operator

has a big enough footprint. Even if operators largely aim to provide the same infrastructure services (VPN connectivity, compute resources based on virtual machines and block storage), inter-operator collaboration tools for providing a service spanning several administrative boundaries are very limited and cumbersome. This makes service development and provisioning very time consuming. For example, having a VPN with end-points in several countries, in order to connect multiple sites of a business (such as a hotel chain), requires contacting several network operators. Such an approach is possible only with significant effort and integration work from the side of the business. This is not only slow, but also inefficient and expensive, since the business also needs to employ networking specialists to do the integration instead of focusing on its core business

Technology fragmentation also represents a major bottleneck internally for an operator. Different networks and different parts of a network may be built as different domains using separate technologies, such as optical or packet switched (with different packet switching paradigms included); having equipment from different vendors; having different control paradigms, etc. Managing and integrating these separate technology domains requires substantial amount of effort, expertise, and time. The associated costs are paid by both network operators and vendors alike, who need to design equipment and develop complex integration features. In addition to technology domains, there are other reasons for having multiple domains within an operator, such as, different geographies, different performance characteristics, scalability, policy or simply historic (e.g., result of a merge or an acquisition). Multiple domains in a network are a necessary and permanent feature however, these should not be a roadblock towards service development and provisioning, which should be fast and efficient.

A solution is needed to deal with both the multi-operator collaboration issue, and address the multi-domain problem within a single network operator. While these two problems are quite different, they also share a lot of common aspects and can benefit from having a number of common tools to solve them.

5. Multi-domain architectural approaches

This section summarizes different architectural options that can be considered to tackle the multi-domain orchestration problem.

5.1. ETSI NFV approaches

Recently, the ETSI NFV ISG has started to look into viable architectural options supporting the placement of functions in different administrative domains. In the document [etsi_nvf_ifa009], different approaches are considered, which we summarize next.

The first option (shown in Figure 3) is based on a split of the NFVO into Network Service Orchestrator (NSO) and Resource Orchestrator (RO). A use case that this separation could enable is the following: a network operator offering its infrastructure to different departments within the same operator, as well as to a different network operator like in cases of network sharing agreements. In this scenario, an administrative domain can be defined as one or more data centers and VIMs, providing an abstracted view of the resources hosted in it.

A service is orchestrated out of VNFs that can run on infrastructure provided and managed by another Service Provider. The NSO manages the lifecycle of network services, while the RO provides an overall view of the resources present in the administrative domain to which it provides access and hides the interfaces of the VIMs present below it.

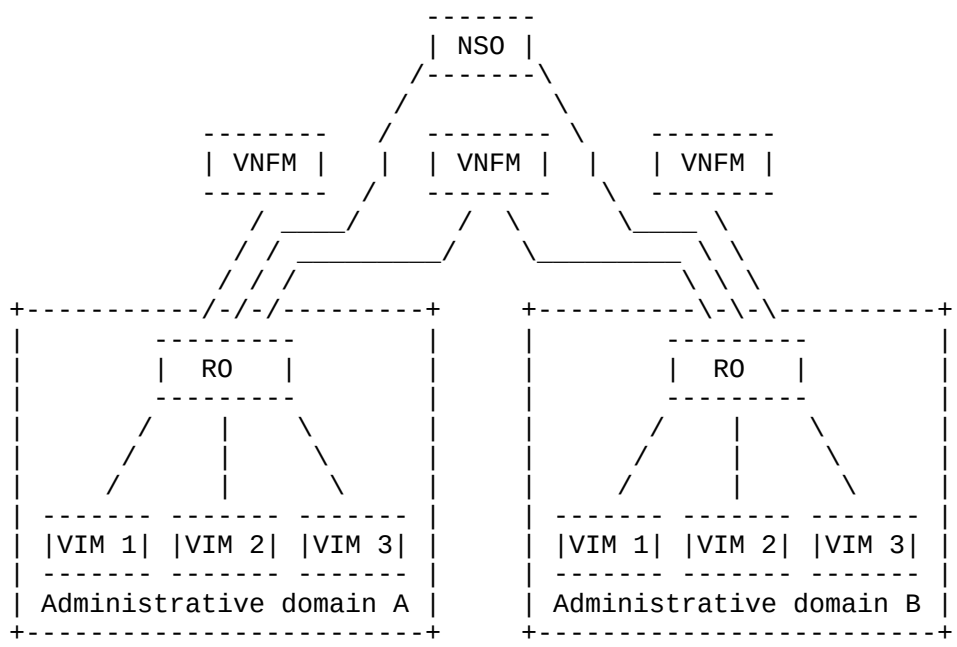


Figure 3: Infrastructure provided using multiple administrative domains (from ETSI GS NFV-IFA 009 V1.1.1)

The second option (shown in Figure 4) is based on having an umbrella NFVO. A use case enabled by this is the following: a Network Operator offers Network Services to different departments within the same operator, as well as to a different network operator like in cases of network sharing agreements. In this scenario, an administrative domain is compose of one or more Datacentres, VIMs, VNFMs (together with their related VNFs) and NFVO, allowing distinct specific sets of network services to be hosted and offered on each.

A top Network Service can include another Network Service. A Network Service containing other Network Services might also contain VNFs. The NFVO in each admin domain provides visibility of the Network Services specific to this admin domain. The umbrella NFVO is providing the lifecycle management of umbrella network services defined in this NFVO. In each admin domain, the NFVO is providing standard NFVO functionalities, with a scope limited to the network services, VNFs and resources that are part of its admin domain.

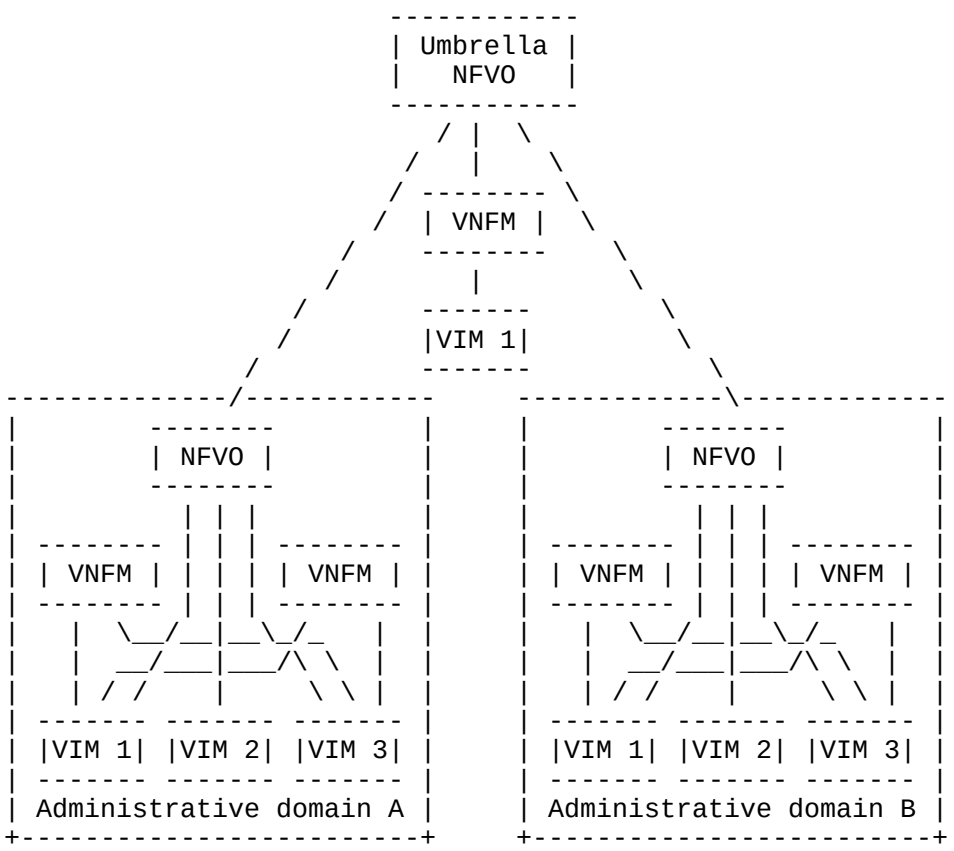


Figure 4: Network services provided using multiple administrative domains (from ETSI GS NFV-IFA 009 V1.1.1)

More recently, ETSI NFV has released a new whitepaper, titled "Network Operator Perspectives on NFV priorities for 5G" [etsi_nvf_whitepaper_5g], which provides network operator perspectives on NFV priorities for 5G and identifies common technical features in terms of NFV. This whitepaper identifies multi-site/ multi-tenant orchestration as one key priority. ETSI highlights the support of Infrastructure as a Service (IaaS), NFV as a Service (NFVaaS) and Network Service (NS) composition in different administrative domains (for example roaming scenarios in wireless networks) as critical for the 5G work.

Related to this, a new Work Item, IFA028, and titled as "Report on architecture options to support multiple administrative domains" has been approved.

5.2. Hierarchical

Considering the potential split of the NFVO into a Network Service Orchestrator (NSO) and a Resource Orchestrator (RO), multi-provider hierarchical interfaces may exist at their northbound APIs. Figure 5 illustrates the various interconnection options, namely:

E/NSO (External NSO): an evolved NFVO northbound API based on Network Service (NS).

E/RO (External RO): VNF-FG oriented resource embedding service. A received VNF-FG that is mapped to the northbound resource view is embedded into the distributed resources collected from southbound, i.e., VNF-FG_in = VNF-FG_out_1 + VNF-FG_out_2 + ... + VNF-FG_out_N, where VNF-FG_out_j corresponds to a spatial embedding to subordinate domain "j". For example, Provider 3's MP-NFVO/RO creates VNF-FG corresponding to its E/RO and E/VIM sub-domains.

E/VIM (External VIM): a generic VIM interface offered to an external consumer. In this case the NFVI-PoP may be shared for multiple consumers, each seeing a dedicated NFVI-PoP. This corresponds to IaaS interface.

I/NSO (Internal NSO): if a Multi-provider NSO (MP-NSO) is separated from the provider's operational NSO, e.g., due to different operational policies, the MP-NSO may need this interface to realize its northbound E/NSO requests. Provider 1 illustrates a scenario the MP-NSO and the NSO are logically separated. Observe that Provider 1's tenants connect to the NSO and MP-NSO corresponds to "wholesale" services.

I/RO (Internal RO): VNF-FG oriented resource embedding service. A received VNF-FG that is mapped to the northbound resource view is embedded into the distributed resources collected from southbound, i.e., VNF-FG_in = VNF-FG_out_1 + VNF-FG_out_2 + ... + VNF-FG_out_N, where VNF-FG_out_j corresponds to a spatial embedding to subordinate domain "j". For example, Provider 1's MP-NFVO/RO creates VNF-FG corresponding to its I/RO and I/VIM sub-domains.

I/VIM (Internal VIM): a generic VIM interface at an NFVI-PoP.

Nfvo-Vim: a generic VIM interface between a (monolithic) NFVO and a VIM.

We would like to explore use-cases and potential benefits for the above multi-provider interfaces as well as to learn how much they may differ from their existing counterparts. For example, are (E/RO, I/RO), (E/NSO, I/NSO), (E/VIM, I/VIM) pairs different?

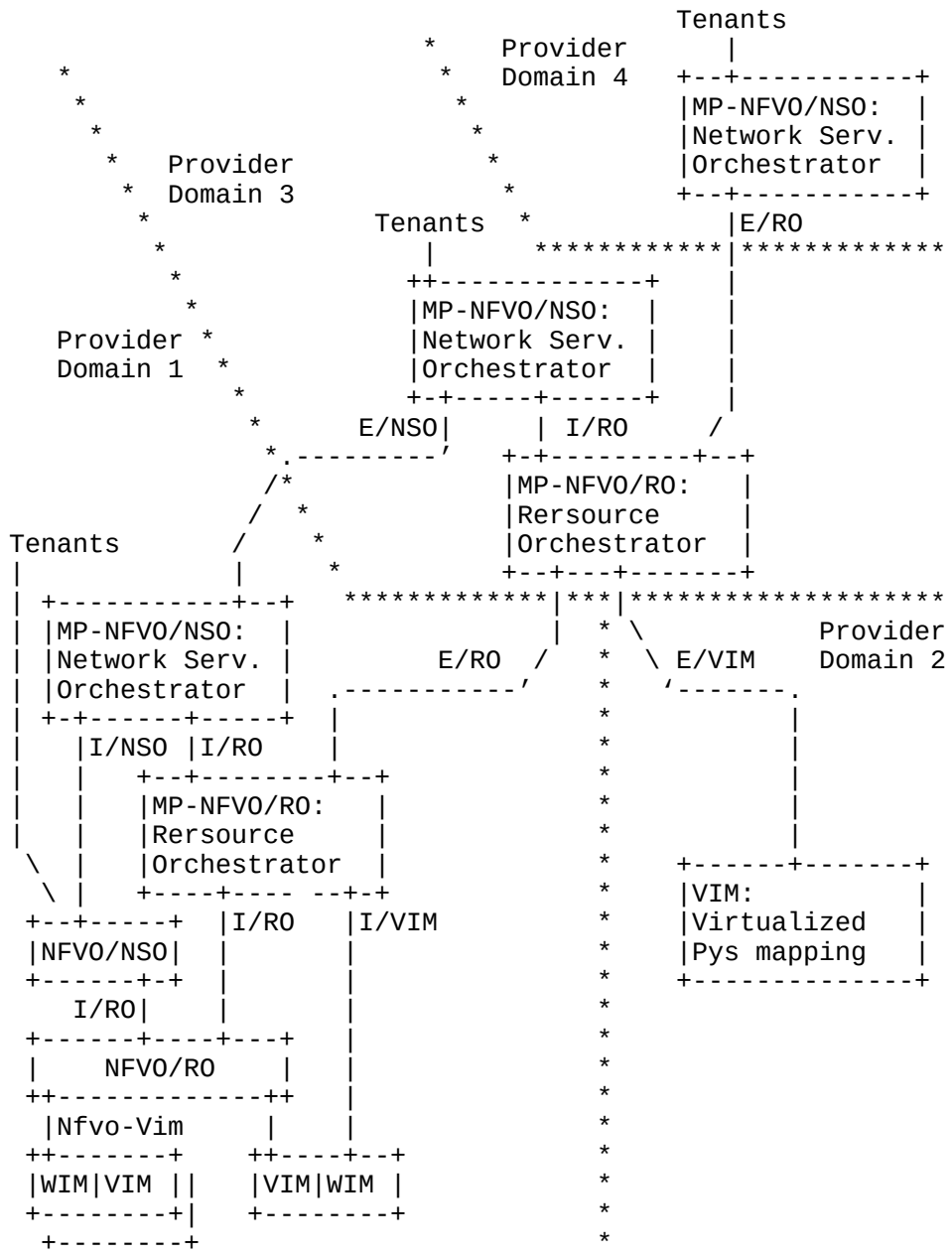


Figure 5: NSO-RO Split: possible multi-provider APIs - an illustration

5.3. Cascading

Cascading is an alternative way of relationship among providers, from the network service point of view. In this case, service decomposition is implemented in a paired basis. This can be extended in a recursive manner, then allowing for a concatenation of cascaded relations between providers.

As a complement to this, from a service perspective, the cascading of two remote providers (i.e., providers not directly interconnected) could require the participation of a third provider (or more) facilitating the necessary communication among the other two. In that sense, the final service involves two providers while the connectivity imposes the participation of more parties at resource level.

6. Virtualization and Control for Multi-Provider Multi-Domain

Orchestration operation in multi-domain is somewhat different from that in a single domain as the assumption in single domain single provider orchestration is that the orchestrator is aware of the entire topology and resource availability within its domain as well as has complete control over those resources. This assumption of technical control cannot be made in a multi domain scenario, furthermore the assumption of the knowledge of the resources and topologies cannot be made across providers. In such a scenario solutions are required that enable the exchange of relevant information across these orchestrators. This exchange needs to be standardized as shown in Figure 6.

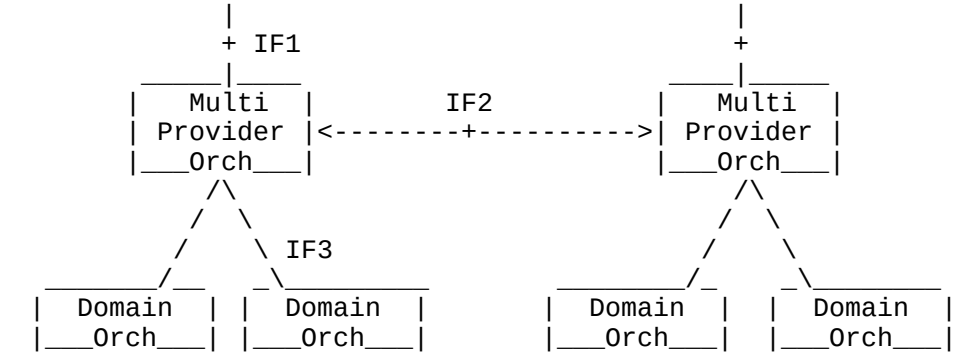


Figure 6: Multi Domain Multi Provider reference architecture

The figure shows the Multi Provider orchestrator exposing an interface 1 (IF1) to the tenant, interface 2 (IF2) to other Multi Provider Orchestrator (MPO) and an interface 3 (IF3) to individual

domain orchestratrators. Each one of these interfaces could be a possible standardization candidate. Interface 1 is exposed to the tennnt who could request his specific services and/or slices to be deployed. Interface 2 is between the orchestrator and is a key interface to enable multi-provider operation. Interface 3 focuses on abstracting the technology or vendor dependent implementation details to support orchestration.

The proposed operation of the MPO follows three main technical steps. First, over interface 2 various functions such as abstracted topology discovery, pricing and service details are detected. Second, once a request for deploying a service is received over interface 1 the Multi Provider Orchestrator evaluates the best orchestrators to implement parts of this request. The request to deploy these parts are sent to the different domain orchestrators over IF2 and IF3 and the acknowledgement that these are deployed in different domain are received back over those interfaces. Third, on receipt of the acknowledgement the slice specific assurance management is started within the MPO. This assurance function collects the appropriate information over IF2 and IF3 and reports the performance back to the tenant over IF1. The assurance is also responsible for detecting any failures in the service and violations in the SLA and recomending to the orchestration engine the reconfiguration of the service or slice which again needs to performed over IF2 and IF3.

Each of the three steps is assigned to a specific block in our high level architecture shown in Figure 7.

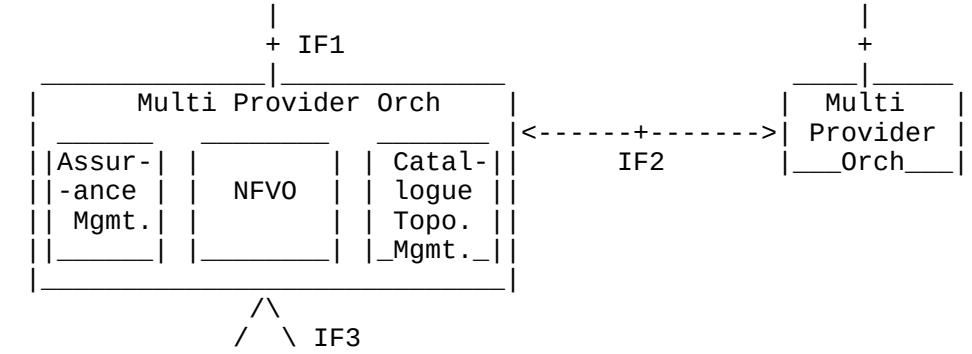


Figure 7: Detailed MPO reference architecture

The catalogue and topology management system is responsible for step 1. It discovers the service as well as the resources exposed by the other domains both on IF2 and IF3. The combination of these services with coverage over the detected topology is provided to the user over IF1. In turn the catalogue and topology management system is also

responsible for exposing the topology and service deployment capabilities to the other domain. The exposure over interface 2 to other MPO maybe abstracted and the mapping of this abstracted view to the real view when requested byt he NFVO.

The NFVO (Network Function Virtualization Orchestrator) is responsible for the second step. It deploys the service or slice as is received from the tenant over IF2 and IF3. It then hands over the deployment decisions to the Assurance managmeent subsystem which use this information to collect the periodic monitoring tickets in step 3. On the other end it is responsible for receiving the request over IF2 to deploy a part of the service, consult with the catalogue and topology management system on the translation of the abstraction to the received request and then for the actual deployment over the domains using IF3. The result of this deployment and the management and control handles to access the deployed slice or service is then returned to the requesting MPO.

The assurance management component periodically studies the collected results to report the overall service performance to the tenant or the requesting MPO as well as to ensure that the service is functioning within the specified parameters. In case of failures or violations the Assurance management system recomends reconfigurations to the NFVO.

6.1. Interworking interfaces

In this section we provide more details on the interworking interfaces of the MPO reference architecture. Each interface IF1, IF2 and IF3 is broken down into several sub-interfaces. Each of them has a clear scope and functionality. [Ed. note: more details will be added in future releases of this document]

For multi provider Network Service orchestration, the Multi-domain Orchestrator (MdO) offers Network Services by exposing an OSS/BSS - NFVO interface to other MPOs belonging to other providers. For multi-provider resource orchestration, the MPO presents a VIM-like view and exposes an extended NFVO - VIM interface to other MPOs. The MPO exposes a northbound sub-interface (IF1-S) through which an MPO customer sends the initial request for services. It handles command and control functions to instantiate network services. Such functions include requesting the instantiation and interconnection of Network Functions (NFs). A sub-interface IF2-S is defined to perform similar operations between MPOs of different administrative domains. A set of sub-interfaces -- IF3-R and IF2-R -- are used to keep an updated global view of the underlying infrastructure topology exposed by domain orchestrators. The service catalogue exposes available services to customers on a sub-interface IF1-C and to other MPO

service operators on sub-interface IF2-C. Resource orchestration related interfaces are broken up to IF2-RC, IF2-RT, IF2-RMon to reflect resource control, resource topology and resource monitoring respectively. Furthermore, the sub-interfaces introduced before are generalised and also used for interfaces IF3 and IF1.

7. IANA Considerations

N/A.

8. Security Considerations

TBD.

9. Acknowledgments

This work is supported by 5G-PPP 5GEx, an innovation action project partially funded by the European Community under the H2020 Program (grant agreement no. 671636). 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 presentation.

10. Informative References

[etsi_nvf_ifa009]
"Report on Architectural Options, ETSI GS NFV-IFA 009 V1.1.1", July 2016.

[etsi_nvf_whitepaper]
"Network Functions Virtualisation (NFV). White Paper 2", October 2014.

[etsi_nvf_whitepaper_5g]
"Network Functions Virtualisation (NFV). White Paper on "Network Operator Perspectives on NFV priorities for 5G"", February 2017.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, S/N
Madrid 28050
Spain

Email: luismiguel.conterasmurillo@telefonica.com

Ishan Vaishnavi
Huawei Technologies Dusseldorf GmBH
Riesstrasse 25,
Munich 80992
Germany

Email: Ishan.vaishnavi@huawei.com

Robert Szabo
Ericsson
Konyves Kaman krt. 11
Budapest, EMEA 1097
Hungary

Phone: +36703135738
Email: robert.szabo@ericsson.com

NFVRG
Internet-Draft
Intended status: Informational
Expires: September 8, 2017

L. Geng
China Mobile

March 7, 2017

Distributed NFV in Scattered Premises
draft-geng-nfvrg-distributed-nfv-00

Abstract

This document introduces the distributed NFV (D-NFV) concept based on potential implementation in scattered locations including customer edge devices and transport network equipments. The motivation of pushing NFV entities from conventional centralized infrastructures to distributed promises is discussed, which are driven by the requirements of high flexibility, low end-to-end latency and faster time-to-market in future network. To better understand the D-NFV concept, examples of D-NFV implementation is introduced. Potential use cases are also described as references for readers. Gaps have also been recognized in the documents in terms of the investigation of appropriate virtualization technologies used in the D-NFV and corresponding management and orchestration challenges.

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 September 8, 2017.

Table of Contents

1. Introduction

As new services such as IoT start to emerge, service provider’s network is required to have higher flexibility, greater security and reliable service quality guarantee from customer end to service provider core. NFV technology has been proved to be an excellent candidate to fulfil these demands for future network. And it has been widely investigated in centralized premises including data centre and mobile core network applications. To further improve overall system flexibility and performance, it is also extremely

interesting to explore how NFV technology can be implemented in scattered network premises.

NFV make use of the virtualization technology to decouple software functions from hardware infrastructures. There is no fundamental limitations on where NFV can be applied to. Some network functions in principle are most efficient when hosted in distributed premises. In this case, it is worth to consider how these functions can be virtualized locally to maintain their efficiency whilst benefit from the flexibility, fast time-to-market deployment and new business model such as VNPaaS that NFV can offer.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Terminology and Abbreviations

The terminology and abbreviations used in this document are defined in this section.

- D-NFV: Distributed NFV. A system architecture where the NFV entities are distributively implemented in scattered network premises.

- NFVI-PoPs: NFV infrastructure points of presence. The location where network functions are realized by VNFs.

4. NFV in Different Points of Presence

In general, NFV provides decoupled software and hardware for vendor-specific network elements. Based on this principle, VNFs are allowed to be located anywhere as long as the corresponding infrastructures support. According to ETSI, proposed points of presence for NFV include customers' premises, central offices, data centres and etc. In principle, VNFs should be placed where they are most cost-effective, providing better efficiency and performances .

4.1. Centralized NFV

At present, most of the NFV deployments are centralized. As an example, the evolving mobile core network is one of the most popular areas where centralized NFV deployment most likely to take place in the near future. It is accelerating its pace in the process of the transition to the next generation NFV-based architecture for 5G. In fact, most of the network functions in core network in form of

conventional vendor-specific network elements are intrinsically centralized. Naturally, the virtualized entities of these network functions are expected to follow the centralized architecture.

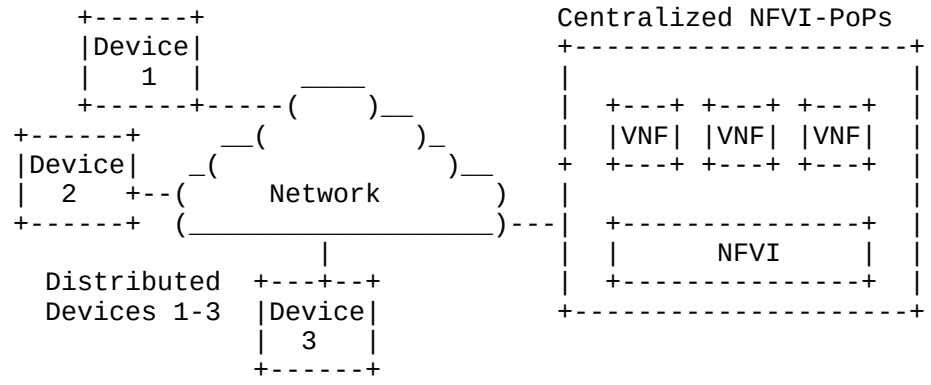


Figure 1

Figure 1 illustrates the scheme diagram of a centralized NFV deployment. In centralized NFV, conventional vendor-specific network elements are realized as VNFs which reside in centralized NFVI-PoPs (i.e. private telecommunication clouds).

The computing and storage hardware resources in the centralized NFV are commonly in the form of server clusters. They are normally pooled and usually span across different physical locations. Network hardware resources (i.e. switches, routers) are essential in centralized NFV to provide connectivities. These include connectivities within and between NFVI-PoPs. Given the powerful computing and storage resources benefited from clusters, centralized NFV is capable of supporting the virtualization of many complex network elements. The COTS used in NFVI also guarantees the system scalability and elastic deployment.

4.2. Distributed NFV

Centralization is not an intrinsic nature of NFV. Many vendor-specific network elements located in scattered premises may also benefit from the implementation of NFV by decoupling software and hardware. Service providers used to replace these equipments or make a full system upgrade on them to deploy new functions and services. With the implementation of NFV, service providers can push new functions and services directly corresponding network elements and end users respectively simply by the deployment of new VNFs.

Many services need local processing provided by network functions are implemented in distributed network elements. These network functions, when virtualized, still make sense to be hosted at the same location for many reasons. Some examples are given as follows.

- o Security. Service requiring end-to-end security has to be implemented from the customer end. VNF for such purposes, i.e. encryption are preferred to be hosted locally.

- o Latency. Mission critical services are very sensitive to latency. Local precessing is preferred to minimize the round trip latency.

- o Resilience. In some scenarios where services are provided remotely in the cloud, customers want their internal networks and services to keep working when there is a network failure. Locally hosted VNFs can work as backups for this purpose.

D-NFV focuses on the scenarios where the NFVI-PoPs locate in scattered premises. Common infrastructures seen in these premises include but are not limited to end-user devices, customer premises equipment and dedicated network equipments in transport and bearer networks.

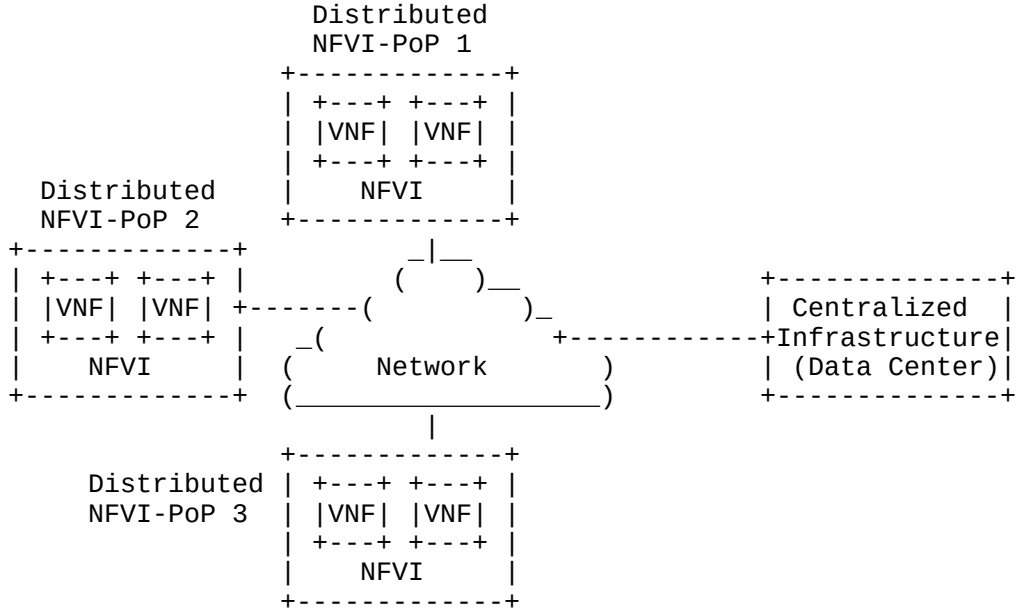


Figure 2

Figure 2 illustrates the scheme diagram of a D-NFV deployment in customer premises. Instead of nested with integrated software and hardware, the CPE provides NFVI for various VNFs. Since the VNFs are decoupled with the hardware resources, service provider can dynamically deploy corresponding VNFs according to performance and customer requirements.

The hardware resources in scattered premises are normally different from that in centralized data centres. Typical examples include SoCs and individual servers. Given the fact that these infrastructures are not designed to be clustered, the network hardware between NFVI-PoPs of D-NFV is not as essential as that of centralized scenario. However, specific services may require network connectivity between NFVI-PoPs to achieve better performances. Network connectivity within NFVI-PoPs in D-NFV may be required depending on the actual design of the VNF entities.Given the limited computing and storage resources, VNFs in the D-NFV should be normally much less resource-hungry than those in centralized NFV.

5. Typical NFVI-PoPs in Distributed NFV

D-NFV focuses on NFV implementations in scattered locations. This section introduces 2 typical examples of NFVI-PoPs including customer edge devices and transport network equipments.

5.1. Distributed NFV in Customer Edge Devices

A customer edge device is the first service-provider-owned device for an end-user to connect to the network and subscribe to specific services. This device is normally purchased by service providers with required functionalities. Accordingly, it normally has well-defined hardware and vendor-specific software.

In residential network, customer edge devices are typically in forms of WiFi routers, with various uplink interface including PON, xDSL and cellular. Service providers used to provide only internet access to residential users and the customer edge devices were rather simple. Recently, many service provider started to provide value-added services such as IPTV, VoIP, home storage, remote download and VPNs. Accordingly, the concept of "intelligent home gateway" is introduced, which enables the residential customer edge device to dynamically implement new services by downloading applications. These application are normally realized as C and Java modules. D-NFV is another way to provide application and hardware decoupling, with extra benefits of better isolation between applications, improved service security, high reliability, managed resource allocation and comprehensive device capability exposure. As IoT services start to emerge in residential market, D-NFV can improve overall deployment

flexibility and generate potential new business model by providing guaranteed isolation, resources and deep capability exposure for different value-added service providers

Other example of customer edge devices include enterprise premise and industrial premise equipment. There are plenty of services which require local implementation and flexible deployment for optimized performance. For example, WAN acceleration and firewalls have best efficiency when they are implemented locally. Meanwhile, low latency services such as manufacture plant control and item tracking in industrial network also require extremely high reliability which need dedicatedly allocated hardware and network resources.

5.2. Distributed NFV in Service Provider Transport and Bearer Networks

The application of NFV in service provider transport network has been investigated mostly in combination of SDN technology. It is interesting to see that NFV as a technology is applied to transport network as a way of implementing the separated control plane in centralized infrastructure. This can be seen as a coordination of SDN and NFV technology with the control plane decoupled and virtualized.

Indeed, as a data plane network equipment, current virtualization technologies are not efficient enough to provide data forwarding performance comparable to network processing chips used in these devices. However, it is attractive to use NFV technology in these network equipment to provide isolated management and control plane. There is great potential for service providers to exploit this technology for a much more flexible management and control model for data plane equipments at a sliced granularity.

6. Use cases of Distributed NFV

In this version, several use cases are listed for general references. Descriptions in detail are subjected to be added according to initial discussion in the group. The author would also like to call for more use cases for D-NFV identified by the community.

6.1. Use Case 1 - VNFaaS and VNPaaS in Residential and Enterprise Network

6.2. Use Case 2 - Mission Critical Services

6.3. Use Case 3 - End-to-end Network Slicing Management

6.4. Use Case 4 - Managed Multiple Provisioning for Network Elements

6.5. Use Case 5 - Elastic VPN

7. Rethinking VNFs in Distributed NFV

In centralized NFV, VNFs are normally virtualized forms of conventional network elements. Sometimes, the network function of a network element may be broken into multiple VNFs for specific implementation considerations. In D-NFVs, VNFs are typically not a full representation of any existing network element. They are more like applications or new services that are pushed to the customer or equipments.

As distributed NFVI-PoP are normally limited in hardware resources, VNFs with complex functionalities are not recommended in these infrastructures. Meanwhile, as VNFs in D-NFV are subjected to be application-specific, it is expected that the variety of VNFs in D-NFV will proportionally grow with the number of services provided through the network. Hence, these VNFs need to have fast time-to-market and adapt to practices like DevOps.

8. Virtualization Technologies in Distributed NFV

The D-NFV may need to consider various virtulization technologies that are different from centralized NFV, as VNFs in D-NFV are expected in much smaller granularities. In this case, container-based virtualization technology may be preferred. This is also due to the potential large number of VNFs and limited hardware resources provided in distributed NFVI-PoPs. Further studies need to be carried out to investigate the appropriated virtualization technologies used in different scenarios of D-NFV

9. Management and Orchestration of Distributed NFV

The management and orchestration of D-NFV need to consider the following difference compared with that of centralized NFV.

- Individually located NFVI and VIM - The nature of D-NFV decide the scattered locations of NFVI-PoPs. In addition, the limited hardware resources are unlikely to support a full MANO implementation in distributed NFVI-PoPs and this is simply not cost-efficient and makes the overall system complicated. Hence a centralized MANO is expected as a feasible solution. This introduce a rather diverted model for the virtualization layer

management where the VIM and NFVI will locate in centralized and distributed infrastructure respectively.

- Massive number of NFVI-PoPs and VNFs - Distributed NFVI-PoPs have a large number in quantity. Taking the residential NFVI-PoP as an example, the number is expected to be millions for a service provide with a fair size business. This does not count the potential industrial and IoT applications which introduce even more. The number of VNFs need to be provisioned can be easily 10-100 times of the NFVI-PoPs. The traditional MANO intrinsically designed for data center applications simply does not fit. It is also too heavy for this purpose - D-NFV may not need such comprehensive resource and network provisioning. The management and orchestration for D-NFV needs to be redesigned.

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

TBA

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

12.2. Informative References

[ETSI-GS-NFV-002] ETSI, "ETSI GS NFV 002", 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf>.

Author’s Address

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

NFVRG
Internet-Draft
Intended status: Informational
Expires: September 1, 2017

R. Gu
S. Hu
China Mobile

February 28, 2017

Control and User Plane Seperation Architecture of Cloud based BNG
draft-gu-nfvrg-cloud-bng-architecture-00

Abstract

This document defines the architecture of Clond-based BNG devices with control plane (CP) and user plane (UP) seperation. Both BNG-CP and BNG-UP are core components for fixed broadband services and deployed seperately at different network layer.

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 September 1, 2017.

Table of Contents

1. Introduction

BNG device is defined as an Ethernet-centric IP edge router, and the aggregation point for the user traffic. It performs Ethernet aggregation and packets forwarding via IP/MPLS, and supports user management, access protocols termination, QoS and policy management, etc.

The basic idea of control plane and user plane seperation is to extract and centralize the user management function of multiple BNG devices forming a separate CP, while UP takes function as router CP and BNG forwarding plane. Thus a BNG is constructed of CP and UP which is benefit in cloud-based BNG with the advantages of resource utilization improvement, resource control centralization, new service rapid provision and so on.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definition of terms

AAA:Authentication Authorization Accounting

BNG:Broadband Network Gateway

CP:Control Plane

DHCP:Dynamic Host Configuration Protocol

MANO:Management and Orchestration

NFV:Network Function Virtualization

PPPoE:Point to Point Protocol over Ethernet

UP: User Plane

4. Cloud-based BNG architecture

There are two parts of functions in traditional BNG: one is user access management function, the other is router function. While in cloud-based BNG, we find out that tearing these two functions apart can make a difference. Actually the user management function can be centralized deployed as a concentrated control plane. The reserved functions such as router functi on and forwarding engine can be deployed in the form of user plane. Thus the Cloud-based BNG architecture is made up of control plane and user plane.

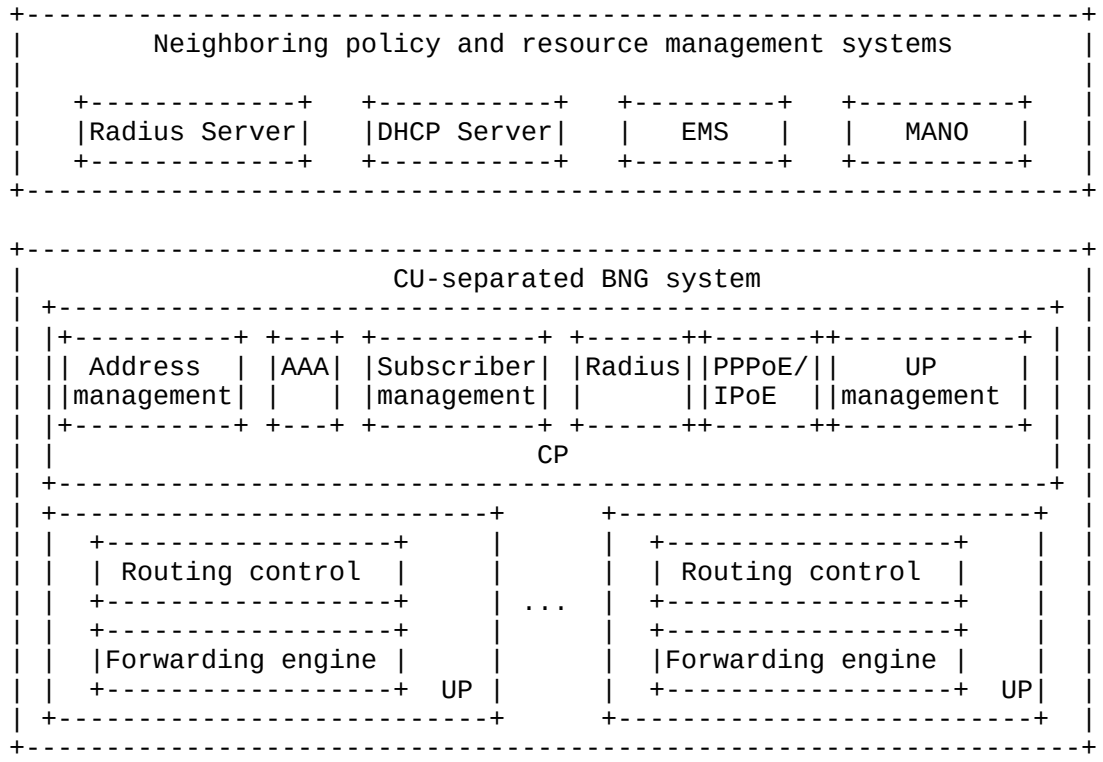


Figure 1: Architecture of C/U Seperation BNG

The CP is a user control management component that supports

(1)address management:unified address pool management

(2)AAA and RADIUS:cooperation with the RADIUS server and others to implement AAA for access users

(3)Subscriber management:user entry management and forwarding policy management

(4)PPPoE/IPoE:process user dialup packets of PPPoE/IPoE

(5)UP management:management of UP interface status, and the setup, deletion, maintenance of channels between CP and UP

The UP is a network edge and user policy implementation component, including

(1)Control plane functions including routing, multicast, and MPLS

(2)forwarding plane functions including traffic forwarding, QoS, and traffic statistics collection

(3)Other functions such as configuration of routing services through EMS Cooperation with the DHCP server or use of the local

Neighboring policy and resource management systems deploys different service systems such as RADIUS server, DHCP server and EMS. Besides NFV infrastructure management system MANO is included in resource mangement systems. All of them have connections with BNG CP.

5. Cloud_BNG use case

In the next generation of Telecom Integrated Cloud (TIC) focusing on content and traffic instead of voice, there can be several layers which we call core TIC layer and edge TIC layer.

Core TIC layer: mainly responsible for control, management, and scheduling functions and carries control-plane NE, centralized media-plane NE, CDN device, and backbone network traffic. BNG-CP is deployed in this layer.

Edge TIC layer: oriented towards major media planes and is mainly responsible for terminating media traffic. BNG-CP is deployed in this layer.

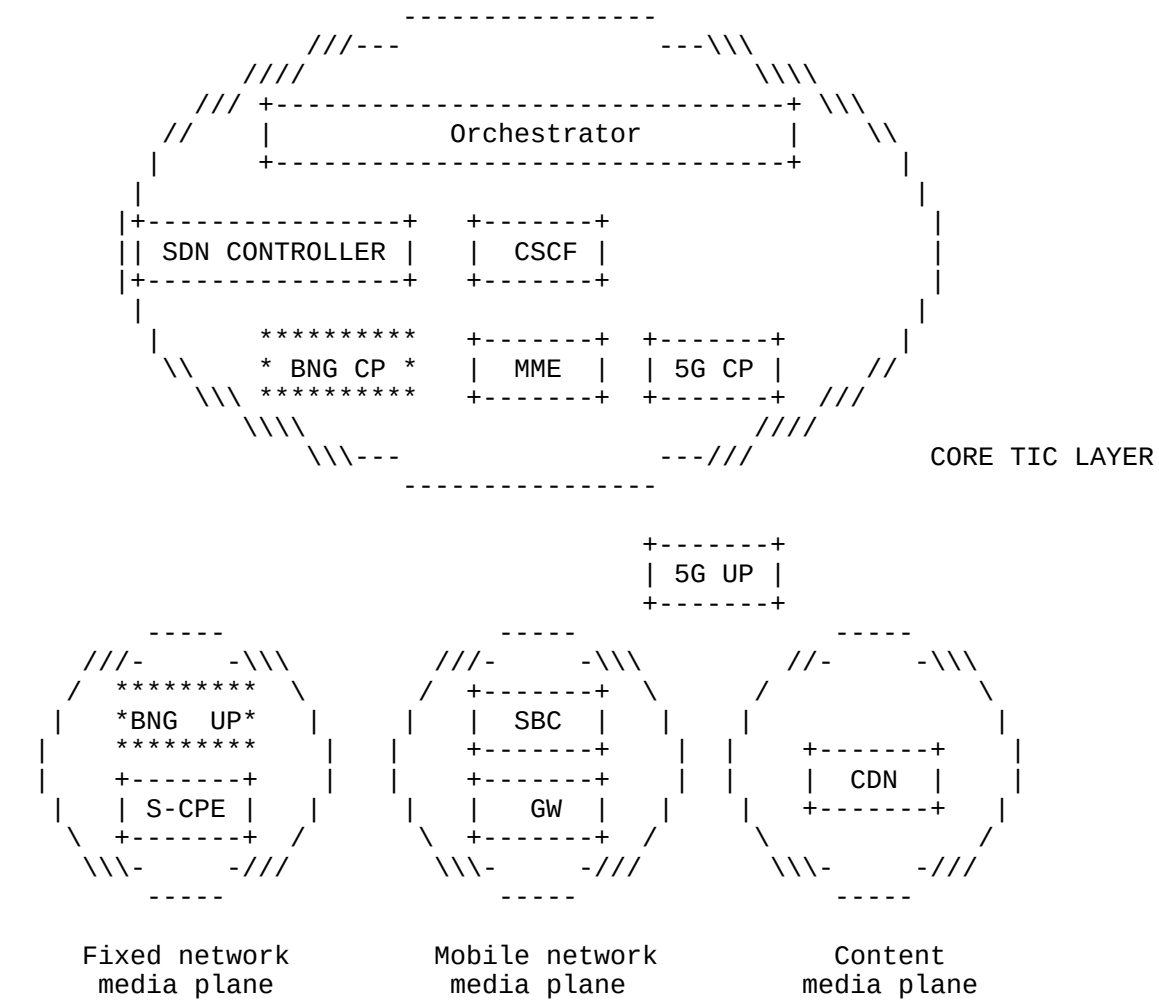


Figure 2: C/U Seperation BNG USE CASE

In the Core TIC layer, BNG CP is deployed including the BNG Service and Subscriber Session Control function which could be virtualized and centralized. The functional components inside the BNG Service and Subscriber Session Control can be implemented as VNFs and run in x86 servers.

In the Edge TIC layer, the routing and forwarding parts could be distributed as BNG UP. Due to high packet processing performance requirements, BNG UP can remain as physical devices. If it’s in some

practical situation, the number of subscribers is relatively small, the BNG UP can be virtualized as well.

6. Cloud_BNG related interfaces

TBD.

7. Security Considerations

None.

8. IANA Considerations

None.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <http://www.rfc-editor.org/info/rfc2234>.

Authors' Addresses

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: gurong_cmcc@outlook.com

Shujun Hu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: hushujun@chinamobile.com.com

NFVRG
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

CJ. Bernardos
UC3M
A. Rahman
InterDigital
JC. Zuniga
SIGFOX
LM. Contreras
TID
P. Aranda
UC3M
P. Lynch
Ixia

March 13, 2017

Network Virtualization Research Challenges
draft-irtf-nfvrg-gaps-network-virtualization-05

Abstract

This document describes open research challenges for network virtualization. Network virtualization is following a similar path as previously taken by cloud computing. Specifically, Cloud computing popularized migration of computing functions (e.g., applications) and storage from local, dedicated, physical resources to remote virtual functions accessible through the Internet. In a similar manner, network virtualization is encouraging migration of networking functions from dedicated physical hardware nodes to a virtualized pool of resources. However, network virtualization can be considered to be a more complex problem than cloud computing as it not only involves virtualization of computing and storage functions but also involves abstraction of the network itself. This document describes current research challenges in network virtualization including guaranteeing quality-of-service, performance improvement, supporting multiple domains, network slicing, service composition, device virtualization, privacy and security. In addition, some proposals are made for new activities in IETF/IRTF that could address some of these challenges.

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/.

Copyright Notice

Table of Contents

1. Introduction

The telecommunications sector is experiencing a major revolution that will shape the way networks and services are designed and deployed for the next few decades. In order to cope with continuously increasing demand and cost, network operators are taking lessons from the IT paradigm of cloud computing. This new approach of virtualizing network functions will enable multi-fold advantages by outsourcing communication services from bespoke hardware in the operator's core network to Commercial off-the-shelf (COTS) equipment distributed across datacenters.

Some of the network virtualization mechanisms that are being considered include: sharing of network infrastructure to reduce costs, virtualization of core servers running in data centers as a way of supporting their load- aware elastic dimensioning, and dynamic energy policies to reduce the electricity consumption.

This document presents research challenges in Network Function Virtualization (NFV) that need to be addressed in order to achieve these goals. The objective of this memo is to document the technical challenges and corresponding current approaches and to expose requirements that should be addressed by future research and standards work.

2. Terminology

The following terms used in this document are defined by the ETSI NVF ISG [etsi_gs_nfv_003], the ONF [onf_tr_521] and the IETF [RFC7665]:

Application Plane - The collection of applications and services that program network behavior.

Control Plane (CP) - The collection of functions responsible for controlling one or more network devices. CP instructs network devices with respect to how to process and forward packets. The control plane interacts primarily with the forwarding plane and, to a lesser extent, with the operational plane.

Forwarding Plane (FP) - The collection of resources across all network devices responsible for forwarding traffic.

Management Plane (MP) - The collection of functions responsible for monitoring, configuring, and maintaining one or more network devices or parts of network devices. The management plane is mostly related to the operational plane (it is related less to the forwarding plane).

NFV Infrastructure (NFVI): totality of all hardware and software components which build up the environment in which VNFs are deployed

NFV Management and Orchestration (NFV-MANO): functions collectively provided by NFVO, VNFM, and VIM.

NFV Orchestrator (NFVO): 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.

OpenFlow protocol (OFP): allowing vendor independent programming of control functions in network nodes.

Operational Plane (OP) - The collection of resources responsible for managing the overall operation of individual network devices.

Physical Network Function (PNF): Physical implementation of a Network Function in a monolithic realization.

Service Function Chain (SFC): for a given service, the abstracted view of the required service functions and the order in which they are to be applied. This is somehow equivalent to the Network Function Forwarding Graph (NF-FG) at ETSI.

Service Function Path (SFP): the selection of specific service function instances on specific network nodes to form a service graph through which an SFC is instantiated.

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

Virtualized Network Function (VNF): implementation of a Network Function that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

Virtualized Network Function Manager (VNFM): functional block that is responsible for the lifecycle management of VNF.

3. Background

3.1. Network Function Virtualization

The ETSI ISG NFV is a working group which, since 2012, aims to evolve quasi-standard IT virtualization technology to consolidate many network equipment types into industry standard high volume servers, switches, and storage. It enables implementing network functions in software that can run on a range of industry standard server hardware and can be moved to, or loaded in, various locations in the network as required, without the need to install new equipment. To date, ETSI NFV is by far the most accepted NFV reference framework and architectural footprint [etsi_nvf_whitepaper_2]. The ETSI NFV framework architecture framework is composed of three domains (Figure 1):

- Virtualized Network Function, running over the NFVI.
- NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualized. NFVI supports the execution of the VNFs.
- NFV Management and Orchestration, which covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs. NFV Management and Orchestration focuses on all virtualization specific management tasks necessary in the NFV framework.

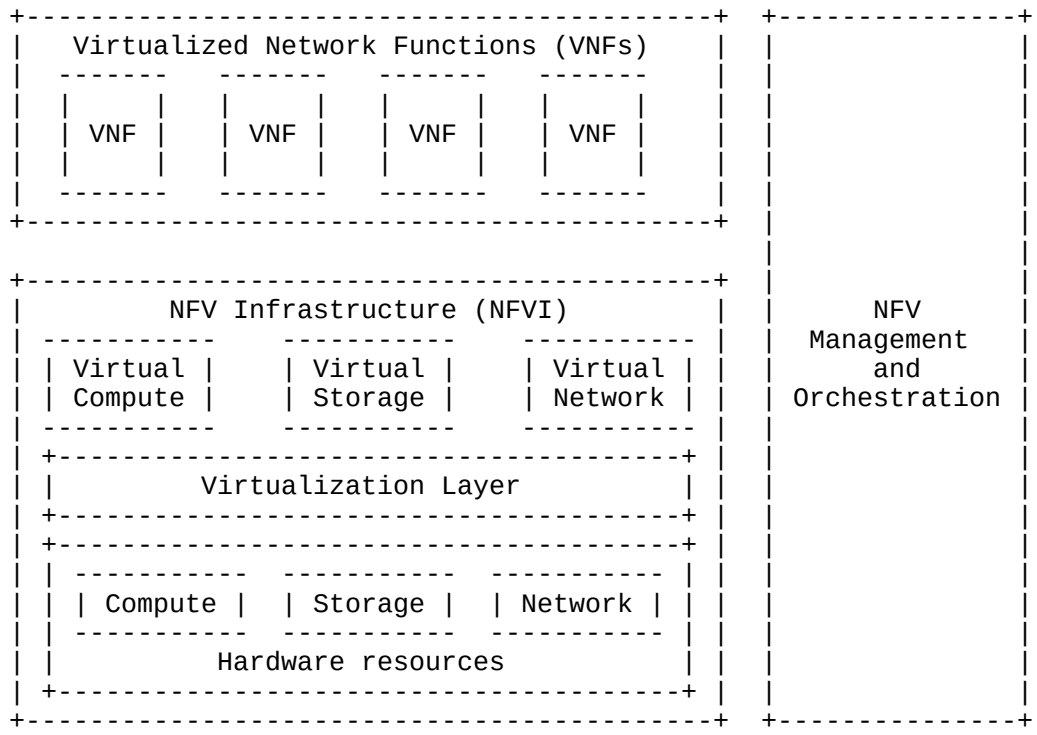


Figure 1: ETSI NFV framework

The NFV architectural framework identifies functional blocks and the main reference points between such blocks. Some of these are already present in current deployments, whilst others might be necessary additions in order to support the virtualization process and consequent operation. The functional blocks are (Figure 2):

- Virtualized Network Function (VNF).
- Element Management (EM).
- NFV Infrastructure, including: Hardware and virtualized resources, and Virtualization Layer.
- Virtualized Infrastructure Manager(s) (VIM).
- NFV Orchestrator.
- VNF Manager(s).
- Service, VNF and Infrastructure Description.

o Operations and Business Support Systems (OSS/BSS).

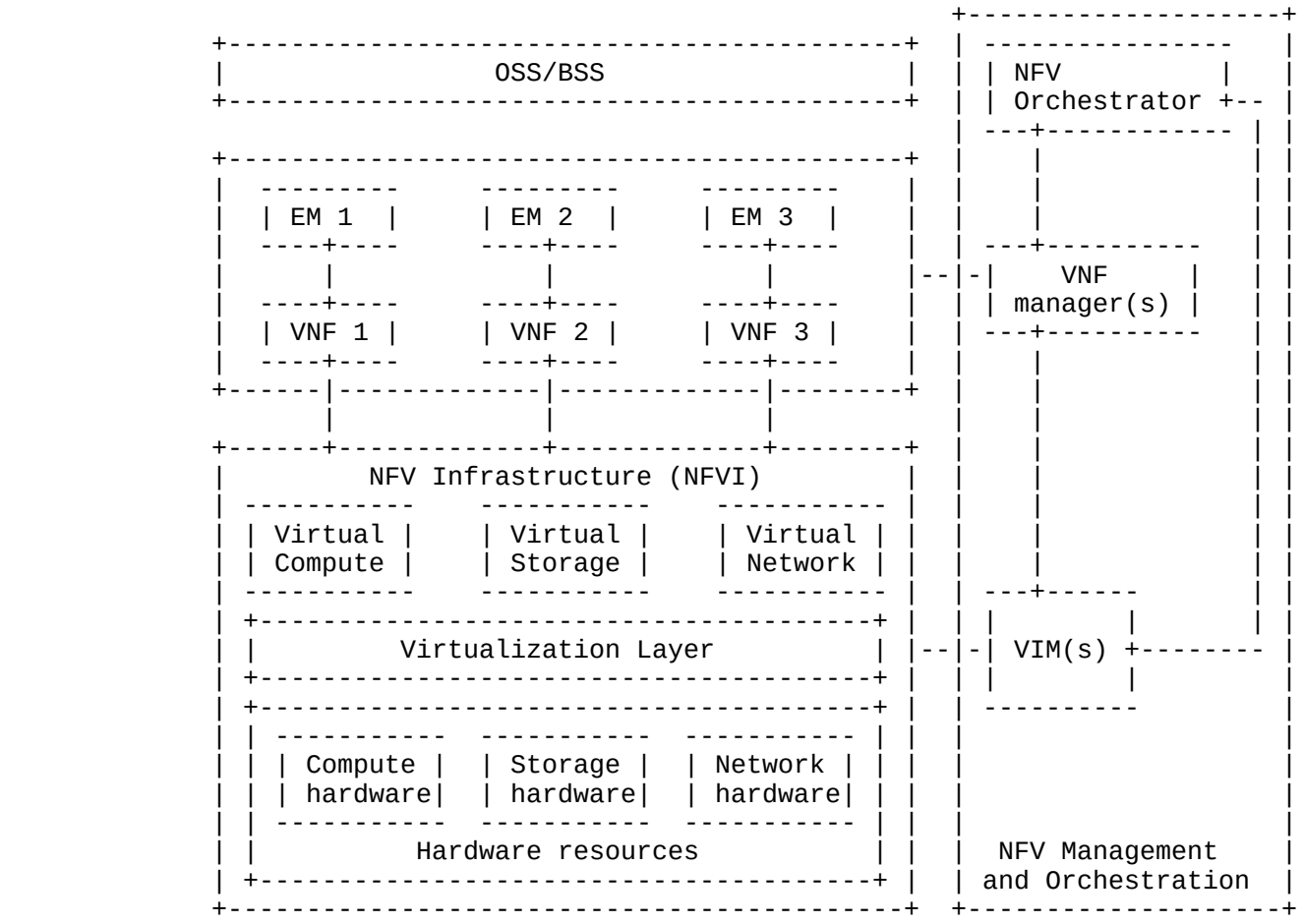


Figure 2: ETSI NFV reference architecture

3.2. Software Defined Networking

The Software Defined Networking (SDN) paradigm pushes the intelligence currently residing in the network elements to a central controller implementing the network functionality through software. In contrast to traditional approaches, in which the network's control plane is distributed throughout all network devices, with SDN the control plane is logically centralized. In this way, the deployment of new characteristics in the network no longer requires of complex and costly changes in equipment or firmware updates, but only a change in the software running in the controller. The main advantage

of this approach is the flexibility it provides operators with to manage their network, i.e., an operator can easily change its policies on how traffic is distributed throughout the network.

The most visible of the SDN protocol stacks is the OpenFlow protocol (OFP), which is maintained and extended by the Open Network Foundation (ONF: https://www.opennetworking.org/). Originally this protocol was developed specifically for IEEE 802.1 switches conforming to the ONF OpenFlow Switch specification. As the benefits of the SDN paradigm have reached a wider audience, its application has been extended to more complex scenarios such as Wireless and Mobile networks. Within this area of work, the ONF is actively developing new OFP extensions addressing three key scenarios: (i) Wireless backhaul, (ii) Cellular Evolved Packet Core (EPC), and (iii) Unified access and management across enterprise wireless and fixed networks.

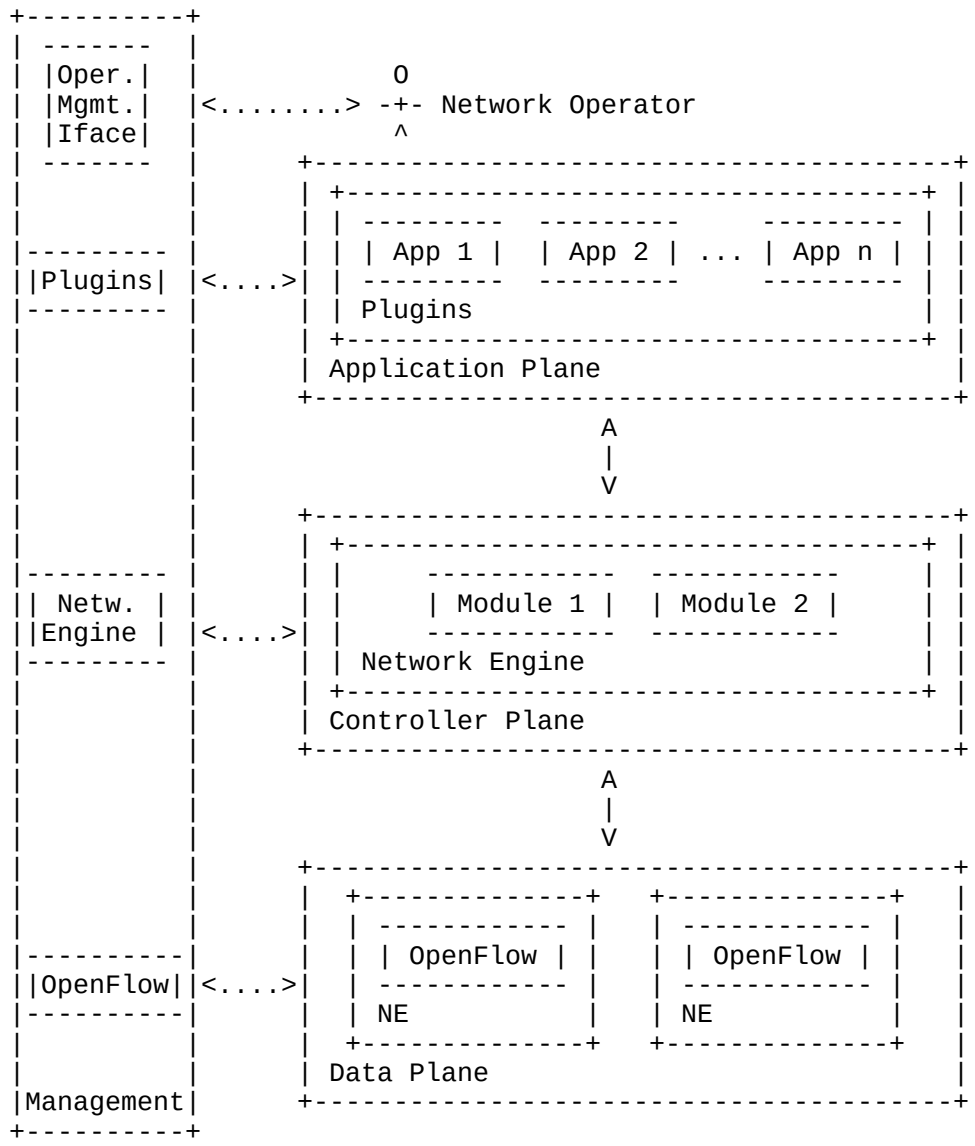


Figure 3: High level SDN ONF architecture

Figure 3 shows the blocks and the functional interfaces of the ONF architecture, which comprises three planes: Data, Controller, and Application. The Data plane comprehends several Network Entities (NE), which expose their capabilities toward the Controller plane via a Southbound API. The Controller plane includes several cooperating modules devoted to the creation and maintenance of an abstracted

resource model of the underneath network. Such model is exposed to the applications via a Northbound API where the Application plane comprises several applications/services, each of which has exclusive control of a set of exposed resources.

The Management plane spans its functionality across all planes performing the initial configuration of the network elements in the Data plane, the assignment of the SDN controller and the resources under its responsibility. In the Controller plane, the Management needs to configure the policies defining the scope of the control given to the SDN applications, to monitor the performance of the system, and to configure the parameters required by the SDN controller modules. In the Application plane, Management configures the parameters of the applications and the service level agreements. In addition to the these interactions, the Management plane exposes several functions to network operators which can easily and quickly configure and tune the network at each layer.

The IRTF Software-Defined Networking Research Group (SDNRG) documented in RFC7426 [RFC7426], a layer model of an SDN architecture, since this has been a controvertial discussion topic: what is exactly SDN? what is the layer structure of the SDN architecture? how do layers interface with each oter? etc.

Figure 4 reproduces the figure included in RFC7426 [RFC7426] to summarize the SDN architecture abstractions in the form of a detailed, high-level schematic. In a particular implementation, planes can be collocated with other planes or can be physically separated.

In SDN, a controller manipulates controlled entities via an interface.Interfaces, when local, are mostly API invocations through some library or system call. However, such interfaces may be extended via some protocol definition, which may use local inter-process communication (IPC) or a protocol that could also act remotely; the protocol may be defined as an open standard or in a proprietary manner.

SDN expands multiple planes: Forwarding, Operational, Control, Management and Applications. All planes mentioned above are connected via interfaces. Additionally, RFC7426 [RFC7426] considers four abstraction layers: the Device and resource Abstraction Layer (DAL), the Control Abstraction Layer (CAL), the Management Abstraction Layer (MAL) and the Network Services Abstraction Layer (NSAL).

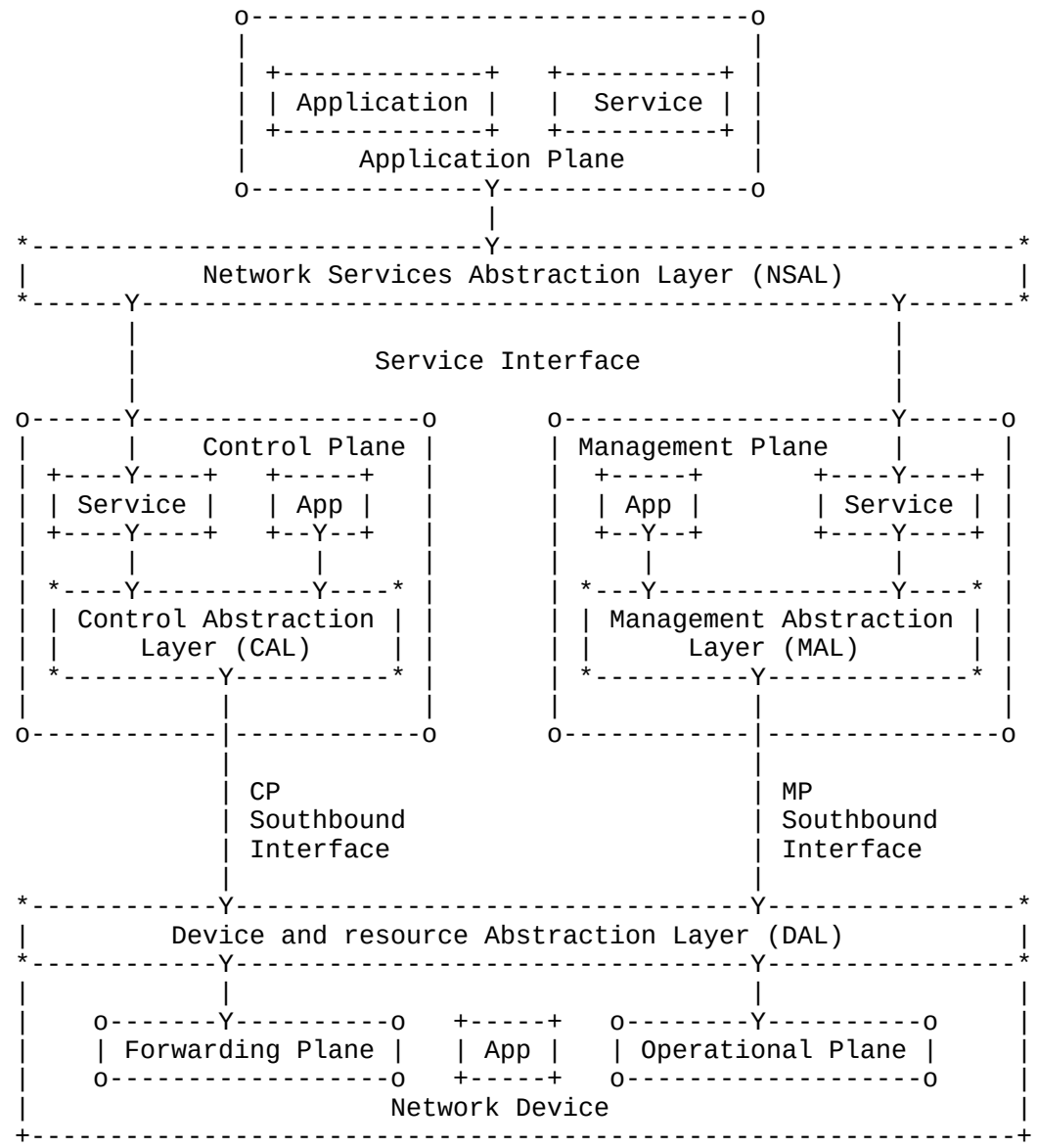


Figure 4: SDN Layer Architecture

3.3. Mobile Edge Computing

Mobile Edge Computing capabilities deployed in the edge of the mobile network can facilitate the efficient and dynamic provision of services to mobile users. The ETSI ISG MEC working group, operative

from end of 2014, intends to specify an open environment for integrating MEC capabilities with service providers networks, including also applications from 3rd parties. These distributed computing capabilities will make available IT infrastructure as in a cloud environment for the deployment of functions in mobile access networks. It can be seen then as a complement to both NFV and SDN.

3.4. IEEE 802.1CF (OmniRAN)

The IEEE 802.1CF Recommended Practice [omniran] specifies an access network, which connects terminals to their access routers, utilizing technologies based on the family of IEEE 802 Standards (e.g., 802.3 Ethernet, 802.11 Wi-Fi, etc.). The specification defines an access network reference model, including entities and reference points along with behavioral and functional descriptions of communications among those entities.

The goal of this project is to help unifying the support of different interfaces, enabling shared network control and use of software defined network (SDN) principles, thereby lowering the barriers to new network technologies, to new network operators, and to new service providers.

3.5. Distributed Management Task Force

The DMTF is an industry standards organization working to simplify the manageability of network-accessible technologies through open and collaborative efforts by some technology companies. The DMTF is involved in the creation and adoption of interoperable management standards, supporting implementations that enable the management of diverse traditional and emerging technologies including cloud, virtualization, network and infrastructure.

There are several DMTF initiatives that are relevant to the network virtualization area, such as the Open Virtualization Format (OVF), for VNF packaging; the Cloud Infrastructure Management Interface (CIM), for cloud infrastructure management; the Network Management (NETMAN), for VNF management; and, the Virtualization Management (VMAN), for virtualization infrastructure management.

3.6. Open Source initiatives

The Open Source community is especially active in the area of network virtualization. We next summarize some of the active efforts:

- OpenStack. OpenStack is a free and open-source cloud-computing software platform. OpenStack software controls large pools of

compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API.

- OpenDayLight. OpenDaylight (ODL) is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multi-vendor networks. It provides a model-driven service abstraction platform that allows users to write apps that easily work across a wide variety of hardware and southbound protocols.

- ONOS. The ONOS (Open Network Operating System) project is an open source community hosted by The Linux Foundation. The goal of the project is to create a software-defined networking (SDN) operating system for communications service providers that is designed for scalability, high performance and high availability.

- OpenContrail. OpenContrail is an Apache 2.0-licensed project that is built using standards-based protocols and provides all the necessary components for network virtualization-SDN controller, virtual router, analytics engine, and published northbound APIs. It has an extensive REST API to configure and gather operational and analytics data from the system.

- OPNFV. OPNFV is a carrier-grade, integrated, open source platform to accelerate the introduction of new NFV products and services. By integrating components from upstream projects, the OPNFV community aims at conducting performance and use case-based testing to ensure the platform’s suitability for NFV use cases. The scope of OPNFV’s initial release is focused on building NFV Infrastructure (NFVI) and Virtualized Infrastructure Management (VIM) by integrating components from upstream projects such as OpenDaylight, OpenStack, Ceph Storage, KVM, Open vSwitch, and Linux. These components, along with application programmable interfaces (APIs) to other NFV elements form the basic infrastructure required for Virtualized Network Functions (VNF) and Management and Network Orchestration (MANO) components. OPNFV’s goal is to increase performance and power efficiency; improve reliability, availability, and serviceability; and deliver comprehensive platform instrumentation.

- OSM. Open Source Mano (OSM) is an ETSI-hosted project to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV. OSM is based on components from previous projects, such Telefonica’s OpenMANO or Canonical’s Juju, among others.

- OpenBaton. OpenBaton is a ETSI NFV compliant Network Function Virtualization Orchestrator (NFVO). OpenBaton was part of the

OpenSDNCore project started with the objective of providing a compliant implementation of the ETSI NFV specification.

Among the main areas that are being developed by the former open source activities that related to network virtualization research, we can highlight: policy-based resource management, analytics for visibility and orchestration, service verification with regards to security and resiliency.

3.7. Internet of Things (IoT)

The Internet of Things (IoT) refers to the vision of connecting a multitude of automated devices (e.g. lights, environmental sensors, traffic lights, parking meters, health and security systems, etc.) to the Internet for purposes of reporting, and remote command and control of the device. This vision is being realized by a multi-pronged approach of standardization in various forums and complementary open source activities. For example, in IETF, support of IoT web services has been defined by an HTTP-like protocol adapted for IoT called CoAP [RFC7252], and lately a group has been studying the need to develop a new network layer to support IP applications over Low Power Wide Area Networks (LPWAN).

Elsewhere, for 5G cellular evolution there is much discussion on the need for supporting virtual "network slices" for the expected massive numbers of IoT devices. A separate virtual network slice is considered necessary for different 5G IoT use cases because devices will have very different characteristics than typical cellular devices like smart phones [ngmn_5G_whitepaper], and the number of IoT devices is expected to be at least one or two orders of magnitude higher than other 5G devices.

4. Network Virtualization Challenges

4.1. Introduction

Network Virtualization is changing the way the telecommunications sector will deploy, extend and operate their networks. These new technologies aim at reducing the overall costs by outsourcing communication services from specific hardware in the operators’ core to server farms scattered in datacenters (i.e. compute and storage virtualization). In addition, the connecting networks are fundamentally affected in the way they route, process and control traffic (i.e. network virtualization).

4.2. Guaranteeing quality-of-service

Guaranteeing a given quality-of-service in an NFV environment is not an easy task. For example, ensuring a guaranteed and stable forwarding data rate has proven not to be straightforward when the forwarding function is virtualized and runs on top of COTS server hardware [openmano_dataplane] [I-D.mlk-nfvrg-nfv-reliability-using-cots] [etsi_nvf_whitepaper_3]. We next identify some of the challenges that this poses.

4.2.1. Virtualization Technologies

The issue of guaranteeing a network quality-of-service is less of an issue for "traditional cloud computing" because the workloads that are treated there are servers or clients in the networking sense and hardly ever process packets. Cloud computing provides hosting for applications on shared servers in a highly separated way. Its main advantage is that the infrastructure costs are shared among tenants and that the Cloud infrastructure provides levels of reliability that can not be achieved on individual premises in a cost-efficient way [intel_10_differences_nfv_cloud]. NFV poses very strict requirements posed in terms of performance, stability and consistency. Although there are some tools and mechanisms to improve this, such as Enhanced Performance Awareness (EPA), SR-IOV, NUMA, DPDK, etc, these are still unsolved challenges. One open research issue is finding out technologies that are different from VM and more suitable for dealing with network functionalities.

Lately, a number of light-weight virtualization technologies including containers, unikernels (specialized VMs) and minimalistic distributions of general-purpose OSes have appeared as virtualization approaches that can be used when constructing an NFV platform. [I-D.natarajan-nfvrg-containers-for-nfv] describes the challenges in building such a platform and discusses to what extent these technologies, as well as traditional VMs, are able to address them.

4.2.2. Metrics for NFV characterization

Another relevant aspect is the need for tools for diagnostics and measurement suited for NFV. There is a pressing need to define metrics and associated protocols to measure the performance of NFV. Specifically, since NFV is based on the concept of taking centralized functions and evolving it to highly distributed SW functions, there is a commensurate need to fully understand and measure the baseline performance of such systems.

The IP Performance Metrics (IPPM) WG defines metrics that can be used to measure the quality and performance of Internet services and

applications running over transport layer protocols (e.g., TCP, UPD) over IP. It also develops and maintains protocols for the measurement of these metrics. While the IPPM WG is a long running WG that started in 1997 it does not have a charter item or active drafts related to the topic of network virtualization. In addition to using IPPM metrics to evaluate the QoS, there is a need for specific metrics for assessing the performance of network virtualization techniques.

The Benchmarking Methodology Working Group (BMWG) is also performing work related to NFV metrics. For example, [I-D.ietf-bmwg-virtual-net] investigates additional methodological considerations necessary when benchmarking VNFs instantiated and hosted in general- purpose hardware, using bare-metal hypervisors or other isolation environments such as Linux containers. An essential consideration is benchmarking physical and virtual network functions in the same way when possible, thereby allowing direct comparison.

As stated in the document [I-D.ietf-bmwg-virtual-net], there is a clear motivation for the work on performance metrics for NFV [etsi_gs_nfv_per_001], that is worth replicating here: "I'm designing and building my NFV Infrastructure platform. The first steps were easy because I had a small number of categories of VNFs to support and the VNF vendor gave HW recommendations that I followed. Now I need to deploy more VNFs from new vendors, and there are different hardware recommendations. How well will the new VNFs perform on my existing hardware? Which among several new VNFs in a given category are most efficient in terms of capacity they deliver? And, when I operate multiple categories of VNFs (and PNFs) *concurrently* on a hardware platform such that they share resources, what are the new performance limits, and what are the software design choices I can make to optimize my chosen hardware platform? Conversely, what hardware platform upgrades should I pursue to increase the capacity of these concurrently operating VNFs?"

Lately, there are also some efforts lately lookinh into VNF benchmarking. The 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 [vnf_benchmarking] [I-D.rorosz-nfvrg-vbaas].

4.2.3. Predictive analysis

On top of diagnostic tools that enable an assessment of the QoS, predictive analyses are required to react before anomalies occur. Due to the SW characteristics of VNFs, a reliable diagnosis framework could potentially enable the prevention of issues by a proper

diagnosis and then a reaction in terms of acting on the potentially impacted service (e.g., migration to a different compute node, scaling in/out, up/down, etc).

4.2.4. Portability

Portability in NFV refers to the ability to run a given VNF on multiple NFVIs, that is, that it is possible to guarantee that the VNF would be able to perform its functions with a high and predictable performance given that a set of requirements on the NFVI resources is met. Therefore, portability is a key feature that, if fully enabled, would contribute to making the NFV environment achieve a better reliability than a traditional system. The fact of running functionality in SW over "commodity" infrastructure should make much easier to port/move functions from one place to another. However this is not yet as ideal as it sounds and there are aspects not fully tackled. The existence of different hypervisors, specific hardware dependencies (e.g., EPA related) or state synchronization aspects are just some examples of trouble-makers for portability purposes.

The ETSI NFV ISG is doing work in relation to portability. [etsi_gs_nfv_per_001] provides a list of minimal features which the VM Descriptor and Compute Host Descriptor should contain for the appropriate deployment of VM Images over an NFVI (i.e. a "telco datacentre"), in order to guarantee high and predictable performance of data plane workloads while assuring their portability. In addition, the document provides a set of recommendations on the minimum requirements which HW and hypervisor should have for a "telco datacentre" suitable for different workloads (data-plane, control-plane, etc.) present in VNFs. The purpose of this document is to provide the list of VM requirements that should be included in the VM Descriptor template, and the list of HW capabilities that should be included in the Compute Host Descriptor (CHD) to assure predictable high performance. ETSI NFV assumes that the MANO Functions will make the mix & match. There are therefore still quite several research challenges to be addressed here.

4.3. Performance improvement

4.3.1. Energy Efficiency

Virtualization is typically seen as a direct enabler of energy savings. Some of the enablers for this that are often mentioned [nfv_sota_research_challenges] are: (i) the multiplexing gains achieved by centralizing functions in data centers reduce overall the energy consumed, (ii) the flexibility brought by network programmability enables to switch off infrastructure as needed in a much easier way. However there is still a lot of room for

improvement in terms of virtualization techniques to reduce the power consumption, such as enhanced hypervisor technologies.

4.3.2. Improved link usage

The use of NFV and SDN technologies can help improving link usage. SDN has shown already that it can greatly increase average link usage (e.g., Google example [google_sdn_wan]). NFV adds more complexity (e.g., due to service function chaining / VNF forwarding drafts) which need to be considered. Aspects like the ones described in [I-D.bagnulo-nfvrg-topology] on NFV data center topology design have to be carefully looked as well.

4.4. Multiple Domains

Market fragmentation has resulted in a multitude of network operators each focused on different countries and regions. This makes it difficult to create infrastructure services spanning multiple countries, such as virtual connectivity or compute resources, as no single operator has a footprint everywhere. Cross-domain orchestration of services over multiple administrations or over multi-domain single administrations will allow end-to-end network and service elements to mix in multi-vendor, heterogeneous technology and resource environments.

For the specific use case of ’Network as a Service’, it becomes even more important to ensure, that Cross Domain Orchestration also takes care of hierarchy of networks and their association, with respect to provisioning tunnels and overlays.

Multi-domain orchestration is currently an active research topic, which is being tackled, among others, by ETSI NFV ISG and the 5GEx project (https://www.5gex.eu/) [I-D.bernardos-nfvrg-multidomain].

4.5. 5G and Network Slicing

From the beginning of all 5G discussions in the research and industry fora, it has been agreed that 5G will have to address much more use cases than the preceding wireless generations, which first focused on voice services, and then on voice and high speed packet data services. In this case, 5G should be able to handle not only the same (or enhanced) voice and packet data services, but also new emerging services like tactile Internet and IoT. These use cases take the requirements to opposite extremes, as some of them require ultra-low latency and higher-speed, whereas some others require ultra-low power consumption and high delay tolerance.

Because of these very extreme 5G use cases, it is envisioned that different radio access networks are needed to better address the specific requirements of each one of the use cases. However, on the core network side, virtualization techniques can allow tailoring the network resources on separate slices, specifically for each radio access network and use case, in an efficient manner.

Network slicing techniques can also allow dedicating resources for even more specific use cases within the major 5G categories. For example, within the major IoT category, which is perhaps the most disrupting one, some autonomous IoT devices will have very low throughput, will have much longer sleep cycles (and therefore high latency), and a battery life thousands of times longer compared to smart phones or some other connected IoT devices that will have almost continuous control and data communications. Hence, it is envisioned that a single virtual core network could be used by slicing separate resources to dedicated radio access networks (RANs) that are better suited for specific use cases.

The actual definition of network slicing is still a sensitive subject, currently under heavy discussion [I-D.gdmb-netslices-intro-and-ps] [I-D.defoy-netslices-3gpp-network-slicing] [ngmn_5G_whitepaper]. Network slicing is a key for introducing new actors in existing market at low cost -- by letting new players rent "blocks" of capacity, if this new market provides performance that are adequate with the application needs (e.g., broadcasting updates to many sensors with satellite broadcasting capabilities). However, more work needs to be done to define how network slicing will impact existing architectures like ETSI NFV, and to define the impacts of network slicing to guaranteeing quality-of-service as described in Section 4.2.

4.5.1. Virtual Network Operators

The widespread of system and network virtualization technologies has conducted to new business opportunities, enlarging the offer of IT resources with virtual network and computing resources, among others. As a consequence, the network ecosystem now differentiates between the owner of physical resources, the Infrastructure Provider (InP), and the intermediary that conforms and delivers network services to the final customers, the Virtual Network Operator (VNO).

VNOs aim to exploit the virtualized infrastructures to deliver new and improved services to their customers. However, current network virtualization techniques offer poor support for VNOs to control their resources. It has been considered that the InP is responsible of the reliability of the virtual resources but there are several

situations in which an VNO requires to gain a finer control on its resources. For instance, dynamic events, such as the identification of new requirements or the detection of incidents within the virtual system, might urge a VNO to quickly reform its virtual infrastructure and resource allocation. However, the interfaces offered by current virtualization platforms do not offer the necessary functions for VNOs to perform the elastic adaptations they require to tackle with their dynamic operation environments.

Beyond their heterogeneity, which can be resolved by software adapters, current virtualization platforms do not have common methods and functions, so it is difficult for the virtual network controllers used by the VNOs to actually manage and control virtual resources instantiated on different platforms, not even considering different InPs. Therefore it is necessary to reach a common definition of the functions that should be offered by underlying platforms to enable such overlay controllers with the possibility of allocate and deallocate resources dynamically and get monitoring data about them.

Such common methods should be offered by all underlying controllers, regardless of being network-oriented (e.g. ODL, ONOS, Ryu) or computing-oriented (e.g. OpenStack, OpenNebula, Eucalyptus). Furthermore, it is also important for those platforms to offer some "PUSH" function to report resource state, avoiding the need for the VNO’s controller to "POLL" for such data. A starting point to get proper notifications within current REST APIs could be to consider the protocol proposed by the WEBPUSH WG.

Finally, in order to establish a proper order and allow the coexistence and collaboration of different systems, a common ontology regarding network and system virtualization should be defined and agreed, so different and heterogeneous systems can understand each other without requiring to rely on specific adatpation mechanisms that might break with any update on any side of the relation.

4.5.2. Extending Virtual Networks and Systems to the Internet of Things

The specific nature of the Internet of Things (IoT) ecosystem, particularly reflected in the Machine-to-Machine (M2M) communications, conducts to the creation of new and highly distributed systems which demand location-based network and computing services. An specific example can be represented by a set of "things" that suddenly require to set-up a firewall to allow external entities to access their data while outsourcing some computation requirements to more powerful systems relying on Cloud-based services. This representative use case exposes important requirements for both NFV and the underlying Cloud infrastructures.

In order to provide the aforementioned location-based functions integrated with highly distributed systems, the so called FOG infrastructures should be able to instantiate VNFs, placing them in the required place, e.g. close to their consumers. This requirement implies that the interfaces offered by virtualization platforms must support the specification of location-based resources, which is a key function in those scenarios. Moreover, those platforms must also be able to interpret and understand the references used by IoT systems to their location (e.g., "My-AP", "5BLDG+2F") and also the specification of identifiers linked to other resources, such as the case of requiring the infrastructure to establish a link between a specific AP and a specific virtual computing node.

4.6. Service Composition

Current network services deployed by operators often involve the composition of several individual functions (such as packet filtering, deep packet inspection, load balancing). These services are typically implemented by the ordered combination of a number of service functions that are deployed at different points within a network, not necessary on the direct data path. This requires traffic to be steered through the required service functions, wherever they are deployed [RFC7498].

For a given service, the abstracted view of the required service functions and the order in which they are to be applied is called a Service Function Chain (SFC), which is called Network Function Forwarding Graph (NF-FG) in ETSI. An SFC is instantiated through selection of specific service function instances on specific network nodes to form a service graph: this is called a Service Function Path (SFP). The service functions may be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.).

Service composition is a powerful tool which can provide significant benefits when applied in a softwarized network environment. There are however many research challenges in this area, as for example the ones related to composition mechanisms and algorithms to enable load balancing and improve reliability. The service composition should also act as an enabler to gather information across all hierarchies (underlays and overlays) of network deployments which may span across multiple operators, for faster serviceability thus facilitating in accomplishing aforementioned goals of "load balancing and improve reliability".

The SFC working group is working on an architecture for service function chaining [RFC7665] that includes the necessary protocols or protocol extensions to convey the Service Function Chain and Service

Function Path information to nodes that are involved in the implementation of service functions and Service Function Chains, as well as mechanisms for steering traffic through service functions.

In terms of actual work items, the SFC WG is has not yet considered working on the management and configuration of SFC components related to the support of Service Function Chaining. This part is of special interest for operators and would be required in order to actually put SFC mechanisms into operation. Similarly, redundancy and reliability mechanisms are currently not dealt with by any WG in the IETF. While this was the main goal of the VNFpool BoF efforts, it still remains unaddressed.

4.7. End-user device virtualization

So far, most of the network softwarization efforts have focused on virtualizing functions of network elements. While virtualization of network elements started with the core, mobile networks architectures are now heavily switching to also virtualize radio access network (RAN) functions. The next natural step is to get virtualization down at the level of the end-user device (i.e., virtualizing a smartphone) [virtualization_mobile_device]. The cloning of a device in the cloud (central or local) bears attractive benefits to both the device and network operations alike (e.g., power saving at the device by offloading computational-heaving functions to the cloud, optimized networking -- both device-to-device and device-to-infrastructure) for service delivery through tighter integration of the device (via its clone in the networking infrastructure). This is being explored for example by the European H2020 ICIRRUS project (www.icirrus-5gnet.eu).

4.8. Security and Privacy

Similar to any other situation where resources are shared, security and privacy are two important aspects that need to be taken into account.

In the case of security, there are situations where multiple vendors will need to coexist in a virtual or hybrid physical/virtual environment. This requires attestation procedures amongst different virtual/physical functions and resources, as well as ongoing external monitoring. Similarly, different network slices operating on the same infrastructure can present security problems, for instance if one slice running critical applications (e.g. support for a safety system) is affected by another slice running a less critical application. In general, the minimum common denominator for security measures on a shared system should be equal or higher than the one required by the most critical application. Multiple and continuous

threat model analysis, as well as DevOps model are required to maintain certain level of security in an NFV system.

On the other hand, privacy in its strictest interpretation, refers to concerns about exposing users of the system to individual threats such as surveillance, identification, stored data compromise, secondary use, intrusion, etc. In this case, the storage, transmission, collection, and potential correlation of information in the NFV system, for purposes not originally intended or not known by the user, should be avoided. This is particularly challenging, as future intentions and threats cannot be easily predicted, and still can be applied for instance on data collected in the past. Therefore, well-known techniques such as data minimization, using privacy features as default, and allowing users to opt in/out should be used to prevent potential privacy issues.

Compared to traditional networks, NFV will result in networks that are much more dynamic (in function distribution and topology) and elastic (in size and boundaries). NFV will thus require network operators to evolve their operational and administrative security solutions to work in this new environment. For example, in NFV the network orchestrator will become a key node to provide security policy orchestration across the different physical and virtual components of the virtualized network. For highly confidential data, for example, the network orchestrator should take into account if certain physical HW of the network is considered more secure (e.g., because it is located in secure premises) than other HW.

Traditional telecom networks typically run under a single administrative domain controlled by an operator. With NFV, it is expected that in many cases, the telecom operator will now become a tenant (running the VNFs), and the infrastructure (NFVI) may be run by a different operator and/or cloud service provider (see also Section 4.4). Thus, there will be multiple administrative domains which will make coordination of security policy more complex. For example, who will be in charge of provisioning and maintaining security credentials such as public and private keys? Also, should private keys be allowed to be replicated across the NFV for redundancy reasons?

On a positive note, NFV will allow better defense against Denial of Service (DoS) attacks because of the distributed nature of the network (i.e. no single point of failure) and the ability to steer (undesirable) traffic quickly [etsi_gs_nfv_sec_001]. Also, NFVs which have physical HW which is distributed across multiple data centers will also provide better fault isolation environments. Especially, if each data center is protected separately via fire walls, DMZs and other network protection techniques.

4.9. Separation of control concerns

NFV environments offer two possible levels of SDN control. One level is the need for controlling the NFVI to provide connectivity end-to-end among VNFs or among VNFs and PNFs (Physical Network Functions). A second level is the control and configuration of the VNFs themselves (in other words, the configuration of the network service implemented by those VNFs), taking profit of the programmability brought by SDN. Both control concerns are separated in nature. However, interaction between both could be expected in order to optimize, scale or influence each other.

Clear mechanisms for such interaction are needed in order to avoid mal-functioning or interference among concerns. These ideas are considered in [etsi_gs_nfv_eve005] and [I-D.irtf-sdnrg-layered-sdn]

4.10. Testing

The impacts of network virtualization on testing can be divided into 3 groups:

1. Changes in methodology.

2. New functionality.

3. Opportunities.

4.10.1. Changes in methodology

The largest impact of NFV is the ability to isolate the System Under Test (SUT). When testing Physical Network Functions (PNF), isolating the SUT means that all the other devices that the SUT communicates with are replaced with simulations (or controlled executions) in order to place the SUT under test by itself. The SUT may be comprised of one or more devices. The simulations use the appropriate traffic type and protocols in order to execute test cases. See Figure 5.

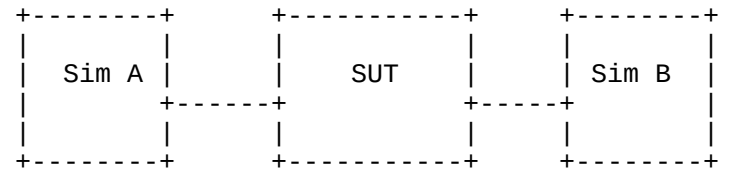


Figure 5: Testing methodology

As shown in Figure 2, NFV provides a common architecture for all functions to use. A VNF is executed using resources offered by the NFVI, which have been allocated using the MANO function. It is not possible to test a VNF by itself, without the entire supporting environment present. This fundamentally changes how to consider the SUT. In the case of a VNF (or multiple VNFs), the SUT is part of a larger architecture which is necessary in order to run the SUTs.

Isolation of the SUT therefore becomes controlling the environment in a disciplined manner. The components of the environment necessary to run the SUTs that are not part of the SUT become the test environment. In the case of VNFs which are the SUT, then the NFVI and MANO become the test environment. The configurations and policies that guide the test environment should remain constant during the execution of the tests, and also from test to test. Configurations such as CPU pinning, NUMA configuration, the SW versions and configurations of the hypervisor, vSwitch and NICs should remain constant. The only variables in the testing should be those controlling the SUT itself. If any configuration in the test environment is changed from test to test, then the results become very difficult, if not impossible, to compare since the test environment behavior may change the results as a consequence of the configuration change.

Testing the NFVI itself also presents new considerations. With a PNF, the dedicated hardware supporting it is optimized for the particular workload of the function. Routing hardware is specially built to support packet forwarding functions, while the hardware to support a purely control plane application (say, a DNS server, or a Diameter function) will not have this specialized capability. In NFV, the NFVI is required to support all types of potentially different workload types.

Testing the NFVI therefore requires careful consideration to what types of metrics are sought. This, in turn, depends on the workload type the expected VNF will be. Examples of different workload types are data forwarding, control plane, encryption, and authentication. All these types of expected workloads will determine the types of metrics that should be sought. For example, if the workload is control plane, then a metric such as jitter is not useful, but dropped packets is critical. In a multi-tenant environment, then the NFVI could support various types of workloads. In this case, testing with a variety of traffic types while measuring the corresponding metrics simultaneously becomes necessary.

4.10.2. New functionality

NFV presents a collection of new functionality in order to support the goal of software networking. Each component on the architecture shown in Figure 2 has an associated set of functionality that allows VNFs to run: onboarding, lifecycle management for VNFs and Networks Services (NS), resource allocation, hypervisor functions, etc.

One of the new capabilities enabled by NFV is VNFFG (VNF Forwarding Graphs). This refers to the graph that represents a Network Service by chaining together VNFs into a forwarding path. In practice, the forwarding path can be implemented in a variety of ways using different networking capabilities: vSwitch, SDN, SDN with a northbound application, and the VNFFG might use tunneling protocols like VXLAN. The dynamic allocation and implementation of these networking paths will have different performance characteristics depending on the methods used. The path implementation mechanism becomes a variable in the network testing of the NSs. The methodology used to test the various mechanisms should largely remain the same, and as usual, the test environment should remain constant for each of the tests, focusing on varying the path establishment method.

Scaling refers to the change in allocation of resources to a VNF or NS. It happens dynamically at run-time, based on defined policies and triggers. The triggers can be network, compute or storage based. Scaling can allocate more resources in times of need, or reduce the amount of resources allocated when the demand is reduced. The SUT in this case becomes much larger than the VNF itself: MANO controls how scaling is done based on policies, and then allocates the resources accordingly in the NFVI. Essentially, the testing of scaling includes the entire NFV architecture components into the SUT.

4.10.3. Opportunities

Softwarization of networking functionality leads to softwarization of test as well. As Physical Network Functions (PNF) are being transformed into VNFs, so have the test tools. This leads to the fact that test tools are also being controlled and executed in the same environment as the VNFs are. This presents an opportunity to include VNF-based test tools along with the deployment of the VNFs supporting the services of the service provider into the host data centers. Tests can therefore be automatically executed upon deployment in the target environment, for each deployment, and each service. With PNFs, this was very difficult to achieve.

This new concept helps to enable modern concepts like DevOps and CI/CD in the NFV environment. Simplistically, DevOps is a process that

combines multiple functions into single cohesive teams in order to quickly produce quality software. It typically relies on also applying the Agile development process, which focusses on (among many things) dividing large features into multiple, smaller deliveries. One part of this is to immediately test the new smaller features in order to get immediate feedback on errors so that if present, they can be immediately fixed and redeployed. The CI/CD (Continuous Integration and Continuous Deployment) pipeline supports this concept. It consists of a series of tools, among which immediate testing is an integral part, to deliver software from source to deployment. The ability to deploy the test tools themselves into the production environment stretches the CI/CD pipeline all the way to production deployment, allowing a range of tests to be executed. The tests can be simple, with a goal of verifying the correct deployment and networking establishment, to the more complex like testing VNF functionality.

5. Technology Gaps and Potential IETF Efforts

Table 1 correlates the open network virtualization research areas identified in this document to potential IETF groups that could address some aspects of them. An example of a specific gap that the group could potentially address is identified in parenthetical beside the group name.

Open Research Area	Potential IETF/IRTF Group
1-Guaranteeing QoS	IPPM WG (Measurements of NFVI)
2-Performance improvement	VNFPOOL BoF (NFV resilience)
3-Multiple Domains	NFVRG
4-Network Slicing	NVO3 WG, NETSLICES bar BoF
5-Service Composition	SFC WG (SFC Mgmt and Config)
6-End-user device virtualization	N/A
7-Security	N/A
8-Separation of control concerns	NFVRG

Table 1: Mapping of Open Research Areas to Potential IETF Groups

6. Mapping to NFVRG Near-Term work items

Table 2 correlates the currently identified NFVRG near-work items to the open network virtualization research areas enumerated in this document. This can help the NFVRG in identifying and prioritizing research topics.

NFVRG Near-Term work item	Open Research Area
1-Policy-based resource management	- Performance improvem.
	- Network Slicing
2-Analytics for visibility & orches.	- Guaranteeing QoS
3-Security and service verification	- Security
4-Reliability and fault detection	- Guaranteeing QoS
5-Service orchestration & lifecycle	- Multiple Domains
	- Network Slicing
	- Service Composition
6-Real-time properties	- Guaranteeing QoS
(other)	- End-user device virt.
	- Separation of control

Table 2: Mapping of NFVRG Near-Term work items to Open Research Areas

7. IANA Considerations

N/A.

8. Security Considerations

This is an informational document, which therefore does not introduce any security threat. Research challenges and gaps related to security and privacy have been included in Section 4.8.

9. Acknowledgments

The authors want to thank Dirk von Hugo, Rafa Marin, Diego Lopez, Ramki Krishnan, Kostas Pentikousis, Rana Pratap Sircar, Alfred Morton, Nicolas Kuhn and Saumya Dikshit for their very useful reviews and comments to the document. Special thanks to Pedro Martinez-Julia, who provided text for the network slicing section.

The work of Carlos J. Bernardos and Luis M. Contreras is partially supported by the H2020-ICT-2014 project 5GEx (Grant Agreement no. 671636).

10. Informative References

[etsi_gs_nfv_003]
ETSI NFV ISG, "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", ETSI GS NFV 003 V1.2.1 NFV 003, December 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf>.

[etsi_gs_nfv_eve005]
ETSI NFV ISG, "Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework", ETSI GS NFV-EVE 005 V1.1.1 NFV-EVE 005, December 2015, <http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf>.

[etsi_gs_nfv_per_001]
ETSI NFV ISG, "Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practises", ETSI GS NFV-PER 001 V1.1.2 NFV-PER 001, December 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/001/01.01.02_60/gs_NFV-PER001v010102p.pdf>.

[etsi_gs_nfv_sec_001]
ETSI NFV ISG, "Network Functions Virtualisation (NFV); NFV Security; Problem Statement", ETSI GS NFV-SEC 001 V1.1.1 NFV-SEC 001, October 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/001/01.01.01_60/gs_NFV-SEC001v010101p.pdf>.

[etsi_nvf_whitepaper_2]
"Network Functions Virtualisation (NFV). White Paper 2", October 2013.

[etsi_nvf_whitepaper_3]
"Network Functions Virtualisation (NFV). White Paper 3", October 2014.

[google_sdn_wan]
"B4: experience with a globally-deployed Software Defined WAN", Proceedings of the ACM SIGCOMM 2013 , August 2013.

[I-D.bagnulo-nfvrg-topology]
Bagnulo, M. and D. Dolson, "NFVI PoP Network Topology: Problem Statement", draft-bagnulo-nfvrg-topology-01 (work in progress), March 2016.

[I-D.bernardos-nfvrg-multidomain]
Bernardos, C., Contreras, L., and I. Vaishnavi, "Multi-domain Network Virtualization", draft-bernardos-nfvrg-multidomain-01 (work in progress), October 2016.

[I-D.defoy-netslices-3gpp-network-slicing]
Foy, X. and A. Rahman, "Network Slicing - 3GPP Use Case", draft-defoy-netslices-3gpp-network-slicing-00 (work in progress), March 2017.

[I-D.gdmb-netslices-intro-and-ps]
Galis, A., Dong, J., kiran.makhijani@huawei.com, k., Bryant, S., Boucadair, M., and P. Martinez-Julia, "Network Slicing - Introductory Document and Revised Problem Statement", draft-gdmb-netslices-intro-and-ps-02 (work in progress), February 2017.

[I-D.ietf-bmwg-virtual-net]
Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", draft-ietf-bmwg-virtual-net-04 (work in progress), August 2016.

[I-D.irtf-sdnrg-layered-sdn]
Contreras, L., Bernardos, C., Lopez, D., Boucadair, M., and P. Iovanna, "Cooperating Layered Architecture for SDN", draft-irtf-sdnrg-layered-sdn-01 (work in progress), October 2016.

[I-D.mlk-nfvrg-nfv-reliability-using-cots]
Mo, L. and B. Khasnabish, "NFV Reliability using COTS Hardware", draft-mlk-nfvrg-nfv-reliability-using-cots-01 (work in progress), October 2015.

[I-D.natarajan-nfvrg-containers-for-nfv]
natarajan.sriram@gmail.com, n., Krishnan, R., Ghanwani, A., Krishnaswamy, D., Willis, P., Chaudhary, A., and F. Huici, "An Analysis of Lightweight Virtualization Technologies for NFV", draft-natarajan-nfvrg-containers-for-nfv-03 (work in progress), July 2016.

[I-D.rorosz-nfvrg-vbaas]
Rosa, R., Rothenberg, C., and R. Szabo, "VNF Benchmark-as-a-Service", draft-rorosz-nfvrg-vbaas-00 (work in progress), October 2015.

[intel_10_differences_nfv_cloud]
Intel, "Discover the Top 10 Differences Between NFV and Cloud Environments", November 2015, <https://software.intel.com/en-us/videos/discover-the-top-10-differences-between-nfv-and-cloud-environments>.

[nfv_sota_research_challenges]
, , , , , and , "Network Function Virtualization: State-of-the-art and Research Challenges", IEEE Communications Surveys & Tutorials Volume: 18, Issue: 1, September 2015.

[ngmn_5G_whitepaper]
"NGMN 5G. White Paper", February 2015.

[omniran] IEEE, "802.1CF Network Reference Model and Functional Description of IEEE 802 Access Network", 802.1cf, Draft 0.4 802.1cf, February 2017.

[onf_tr_521]
ONF, "SDN Architecture, Issue 1.1", ONF TR-521 TR-521, February 2016, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf>.

[openmano_dataplane]
Telefonica I+D, "OpenMANO: The Dataplane Ready Open Source NFV MANO Stack", March 2015, <https://www.ietf.org/proceedings/92/slides/slides-92-nfvrg-7.pdf>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <http://www.rfc-editor.org/info/rfc7426>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <http://www.rfc-editor.org/info/rfc7498>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

[virtualization_mobile_device] "Virtualization of Mobile Device User Experience", Patent US 9.542.062 B2 , January 2017.

[vnf_benchmarking] FEEC/UNICAMP, FEEC/UNICAMP, and Ericsson, "A VNF Testing Framework Design, Implementation and Partial Results", November 2016, <https://www.ietf.org/proceedings/97/slides/slides-97-nfvrg-06-vnf-benchmarking-00.pdf>.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Akbar Rahman
InterDigital Communications, LLC
1000 Sherbrooke Street West, 10th floor
Montreal, Quebec H3A 3G4
Canada

Email: Akbar.Rahman@InterDigital.com
URI: http://www.InterDigital.com/

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: j.c.zuniga@ieee.org
URI: http://www.sigfox.com/

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, S/N
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

Pedro Aranda
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Email: pedroandres.aranda@uc3m.es

Pierre Lynch
Ixia

Email: plynch@ixiacom.com

nfvrg
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

R. Szabo, Ed.
Ericsson
S. Lee, Ed.
ETRI
N. Figueira
Brocade

October 30, 2016

Policy-Based Resource Management
draft-irtf-nfvrg-policy-based-resource-management-02

Abstract

abstract to be defined

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 May 3, 2017.

Copyright Notice

Table of Contents

1. Introduction

NFV "Point of Presence" (PoP) will be likely constrained in compute and storage capacity. Since practically all NFV PoPs are foreseen to be distributed, inter-datacenter network capacity is also a constraint. Additionally, energy is also a constraint, both as a general concern for NFV operators, and in particular for specific-

purpose NFV PoPs such as those in mobile base stations. This draft focuses on the optimized resource management and workload distribution based on policy to address such contraints.

1.1. Scope

For the first version of the draft, only the research group currently adopted drafts (i.e., [I-D.norival-nfvrg-nfv-policy-arch], [I-D.irtf-nfvrg-resource-management-service-chain], and [I-D.unify-nfvrg-recursive-programming]) are considered as inputs to this document. The initial goal is to summarize these inputs and to assess gaps and open questions.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definitions

This document uses the terms of [ETSI-NFV-TERM]:

- MANO - Management and Orchestration: Describes the architecture framework to manage NFVI and orchestrate the allocation of resources needed by the NSs and VNFs.

- NF - Network Functions: A functional building block within a network infrastructure, which has well-defined external interfaces and a well-defined functional behavior.

- NFV Framework: The totality of all entities, reference points, information models and other constructs defined by the specifications published by the ETSI ISG NFV.

- NFVI - NFV Infrastructure: The NFV-Infrastructure is the totality of all hardware and software components which build up the environment in which VNFs are deployed.

- NFVI-PoP: A location or point of presence that hosts NFV infrastructure

- NFVO - Network Function Virtualization Orchestrator: The NFV Orchestrator is in charge of the network wide orchestration and management of NFV (infrastructure and software) resources, and realizing NFV service topology on the NFVI.

- NS - Network service: A composition of network functions and defined by its functional and behavioural specification.

- VNF - Virtualized Network Function: An implementation of an NF that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

- VNF-FG - VNF Forwarding Graph: A NF forwarding graph where at least one node is a VNF.

Additionally, we use the following terms:

- NFP - Network Forwarding Path: The sequence of hardware/software switching ports and operations in the NFV network infrastructure as configured by management and orchestration that implements a logical VNF forwarding graph "link" connecting VNF "node" logical interfaces.

- Virtual Link: A set of connection points along with the connectivity relationship between them and any associated target performance metrics (e.g. bandwidth, latency, QoS). The Virtual Link can interconnect two or more entities (VNF components, VNFs, or PNFs).

- Scaling: Ability to dynamically extend/reduce resources granted to the Virtual Network function (VNF) as needed.

- NFVIaaS: NFV infrastructure as a service to other SP customers.

- SDN: Software Defined Networking.

- BSS: Business Support Systems

- OSS: Operation Support Systems

- DC: Data Center

- VM: Virtual machine

4. Requirements

tbd

5. Architecture Considerations

5.1. MANO Architecture

According to the ETSI MANO framework [ETSI-NFV-MANO], an NFVO is split into two functions (see Figure 1):

- The orchestration of NFVI resources across multiple VIMs, fulfilling the Resource Orchestration functions. The NFVO uses the Resource Orchestration functionality to provide services that support accessing NFVI resources in an abstracted manner independently of any VIMs, as well as governance of VNF instances sharing resources of the NFVI infrastructure

- The lifecycle management of Network Services, fulfilling the network Service Orchestration functions.

Similarly, a VIM is split into two functions (see Figure 1):

- Orchestrating the allocation/upgrade/release/reclamation of NFVI resources (including the optimization of such resources usage), and

- managing the association of the virtualised resources to the physical compute, storage, networking resources.

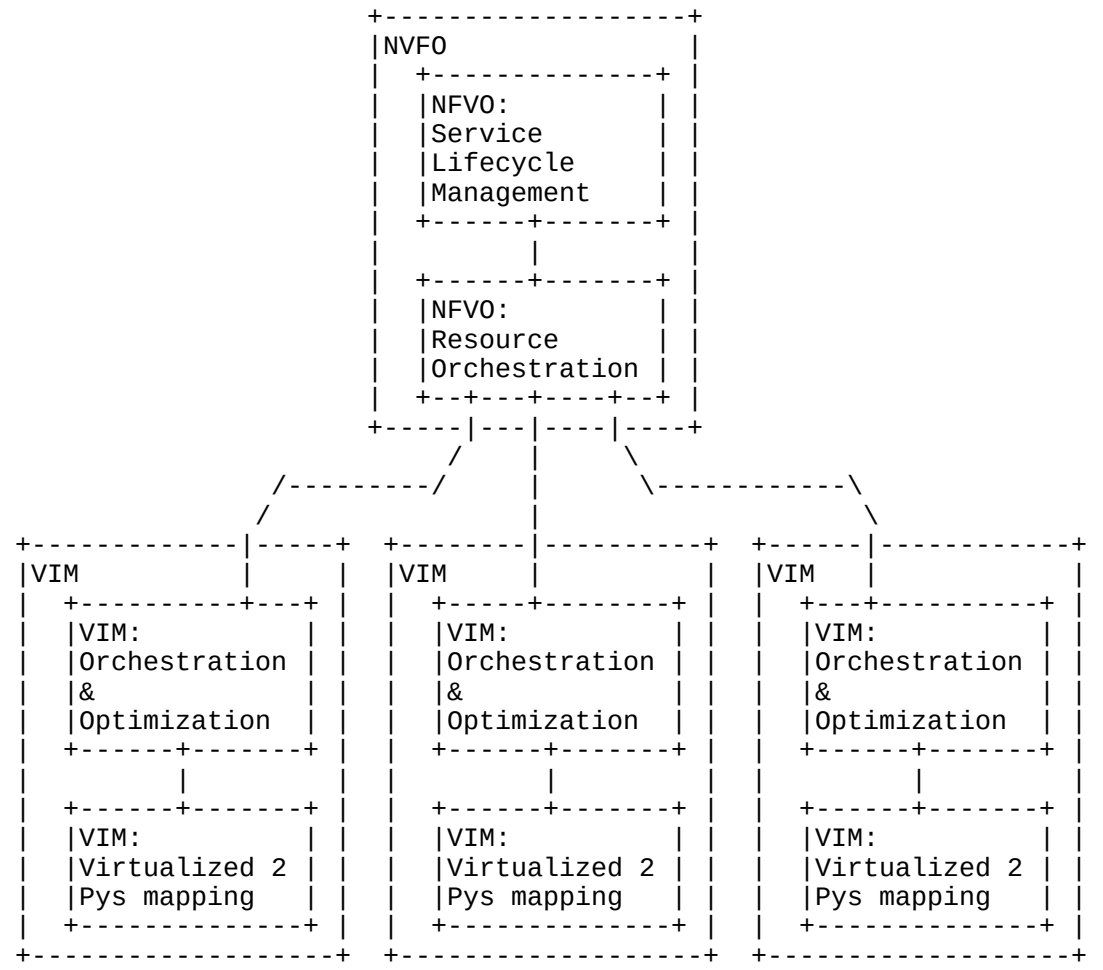


Figure 1: Functional decomposition of the NFVO and the VIM according to the ETSI MANO

In Figure 2 we show various policies mapped to the MANO architecture (see Section 5.2 for more dicussions on policies in the MANO architeture):

- Tenant Policies: Tenant policies exist whenever a domain offers a virtualization service to more than one consumer. User tenants may exists at the northbound of the NFVO. Additionally, if a VIM exposes resource services to more than one NFVO, then each NFVO may appear as a tenant (virtualization consumer) at the northbound of the VIM.

- Wherever virtualization services are produced or consumed corresponding export and import policies may exist. Export policies govern the details of resources, capabilities, costs, etc. exposed to consumers. In turn, consumers (tenants) apply import policies to filter, tweak, annotate resources and services received from their southbound domains. An entity may at the same time consume and produce virtualization services hence apply both import and export policies.

- Operational policies support the business logic realized by the domain’s ownership. They are often associated with Operations or Business Support Systems (OSS or BSS) and frequently determine operational objectives like energy optimization, utilization targets, offered services, charging models, etc. Operational policies may be split according to different control plane layers, for example, i) lifecycle and ii) resource management layers within the NFVO.

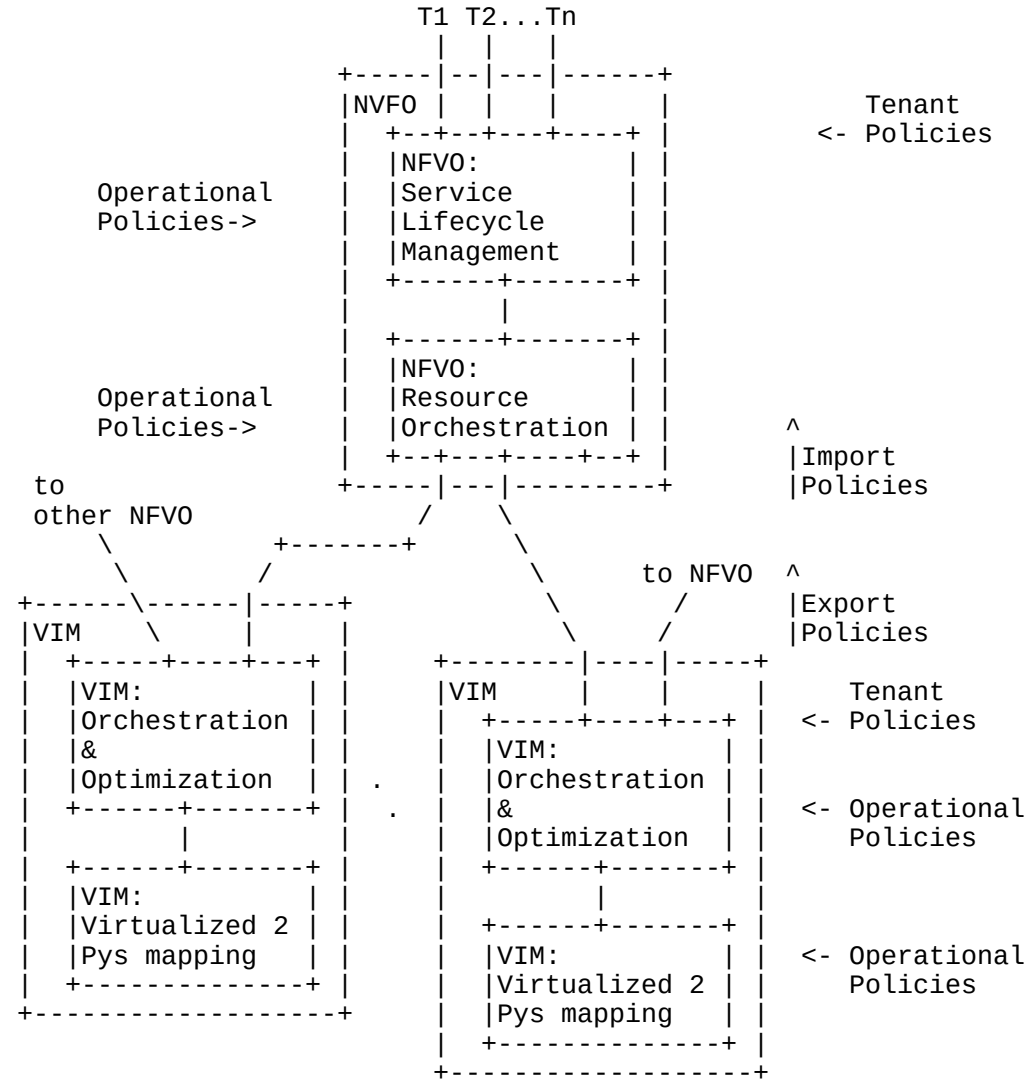


Figure 2: Policies within the MANO framework

5.2. Policies in the MANO Architecture

The current industry work in the area of policy for NFV is mostly considered in the framework of general cloud services, and typically focused on individual subsystems and addressing very specific use cases or environments. For example, [ETSI-NFV-WHITE-PAPER] addresses network subsystem policy for network virtualization, [ODL-GB-POLICY] and [ODL-NIC-PROJECT] are open source projects in the area of network

policy as part of the OpenDaylight [ODL-SDN-CONTROLLER] software defined networking (SDN) controller framework, [RFC3060] specifies an information model for network policy, [VM-HOSTING-NET-CLUSTER] focuses on placement and migration policies for distributed virtual computing, [OPENSTACK-CONGRESS] is an open source project proposal in OpenStack [OPENSTACK] to address policy for general cloud environments.

A policy framework applicable to the MANO architure must consider NFV services from the perspective of overall orchestration requirements for services involving multiple subsystems (e.g., Figure 1 and Figure 2).

While this document discusses policy atributes as applicable to the MANO architecture, the general topic of policy has already been intensively studied and documented on numerous publications over the past 10 to 15 years (see [RFC3060], [POLICY-FRAMEWORK-WG], [RFC3670], [RFC3198], and [CERI-DATALOG] to name a few). This document's purpose is to discuss and document a policy framework applicable to the MANO architecture using known policy concepts and theories to address the unique requirements of NFV services including multiple PoPs and networks forming hierarchical domain architectures [SDN-MULTI-DOMAIN].

With the above goals, this document analyses "global versus local policies" (Section 5.3), a "hierarchical policy framework" (Section 5.4) to address the demanding and growing requirements of NFV environments, a "policy pub/sub bus in the hierarchical framework" (Section 5.5), "policy intent versus subsystem actions" (Section 5.6), "static versus dynamic versus autonomic policies" (Section 5.7), "policy conflict detection and resolution" (Section 5.8), and "soft versus hard policy constraints" (Section 5.9), which can be relevant to resource management in service chains [RESOURCE-MGMT-SERVICE-CHAIN].

5.3. Global vs Local Policies

Some policies may be subsystem specific in scope, while others may have broader scope and interact with multiple subsystems. For example, a policy constraining certain customer types (or specific customers) to only use certain server types for VNF or Virtual Machine (VM) deployment would be within the scope of the compute subsystem. A policy dictating that a given customer type (or specific customers) must be given "platinum treatment" could have different implications on different subsystems. As shown in Figure 8, that "platinum treatment" could be translated to servers of a given performance specification in a compute subsystem and storage of a given performance specification in a storage subsystem.

Policies with broader scope, or global policies, would be defined outside affected subsystems and enforced by a global policy engine (Figure 3), while subsystem-specific policies or local policies, would be defined and enforced at the local policy engines of the respective subsystems.

Examples of sub-system policies can include thresholds for utilization of sub-system resources, affinity/anti-affinity constraints with regard to utilization or mapping of sub-system resources for specific tasks, network services, or workloads, or monitoring constraints regarding under-utilization or over-utilization of sub-system resources.

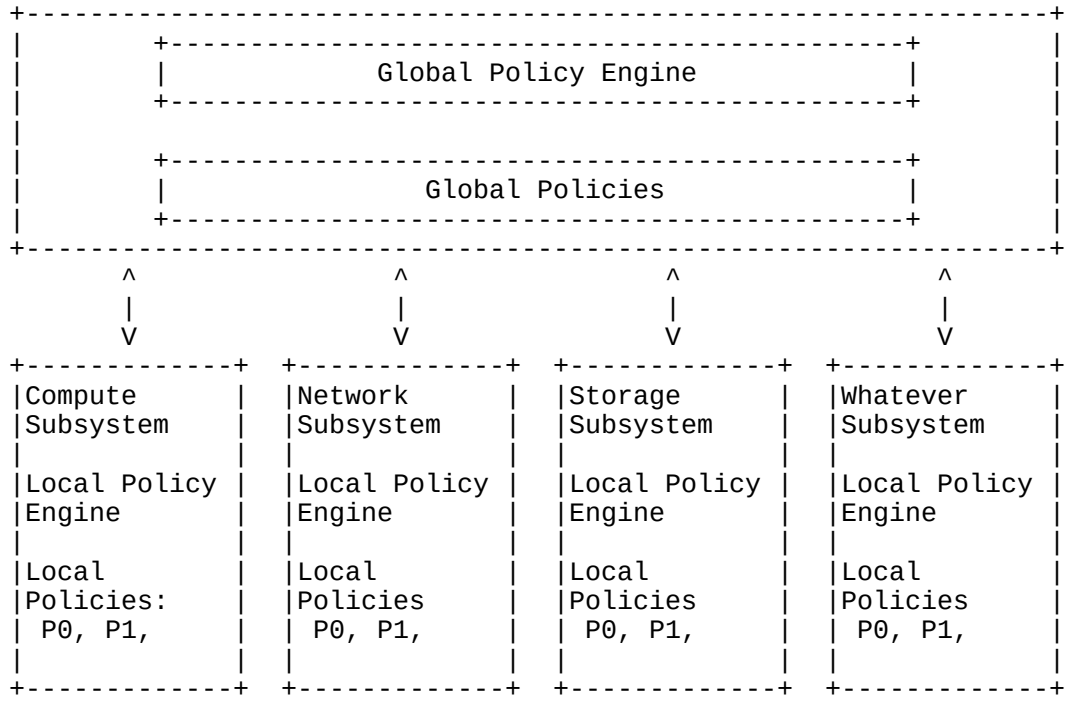


Figure 3: Global versus Local Policy Engines

5.4. Hierarchical Policy Framework

So far, we have referenced compute, network, and storage as subsystems examples. However, the following subsystems may also support policy engines and subsystem specific policies:

- SDN Controllers, e.g., OpenDaylight [ODL-SDN-CONTROLLER].

- OpenStack [OPENSTACK] components such as, Neutron, Cinder, Nova, and etc.
- Directories, e.g., LDAP, ActiveDirectory, and etc.
- Applications in general, e.g., standalone or on top of OpenDaylight or OpenStack.
- Physical and virtual network elements, e.g., routers, firewalls, application delivery controllers (ADCs), and etc.
- Energy subsystems, e.g., OpenStack Neat [OPENSTACK-NEAT].

Therefore, a policy framework may involve a multitude of subsystems. Subsystems may include other lower level subsystems, e.g., Neutron [OPENSTACK-NEUTRON] would be a lower level subsystem in the OpenStack subsystem. In other words, the policy framework is hierarchical in nature, where the policy engine of a subsystem may be viewed as a higher level policy engine by lower level subsystems. In fact, the global policy engine in Figure 3 could be the policy engine of a Data Center subsystem and multiple Data Center subsystems could be grouped in a region containing a region global policy engine. In addition, one could define regions inside regions, hierarchically, as shown in Figure 4.

Metro and wide-area network (WAN) used to interconnect data centers would also be independent subsystems with their own policy engines.

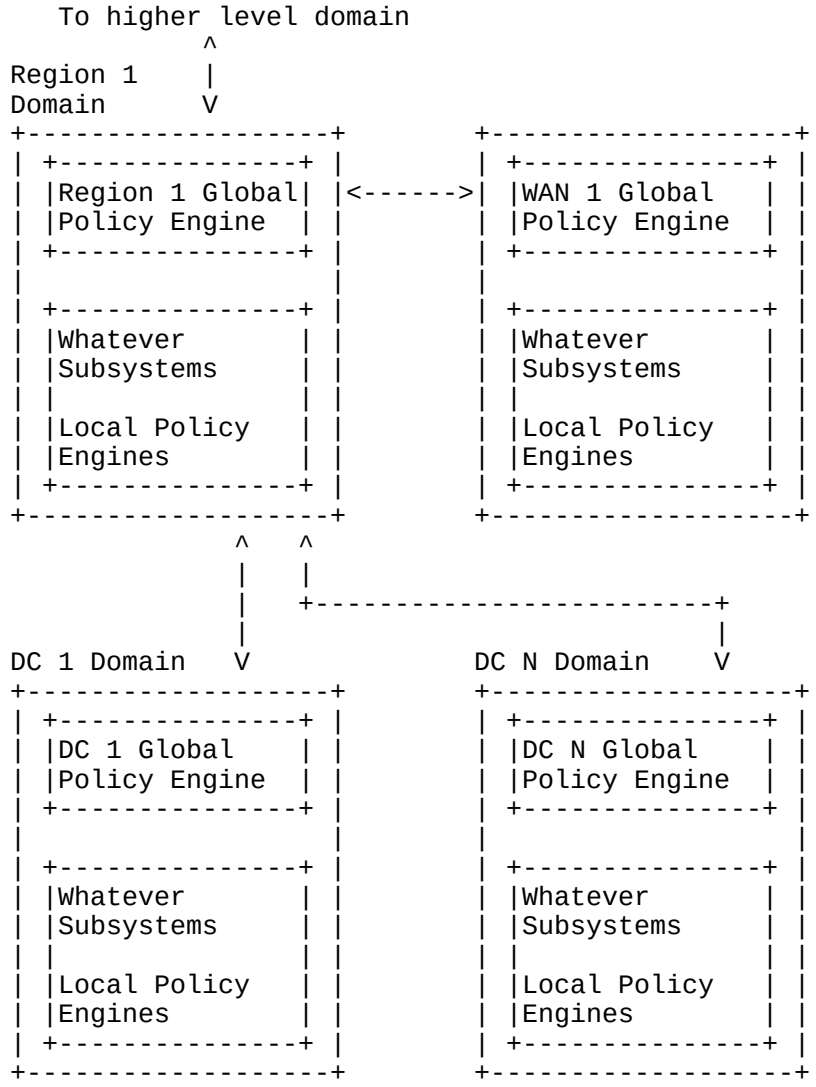


Figure 4: A Hierarchical Policy Framework

5.4.1. Mapping to Hierarchical Resource Orchestration

If the MANO framework is extended to multi layer hierarchies [I-D.unify-nfvrg-recursive-programming], then a potential mapping of the hierarchical policies to the MANO architecture is shown in Figure 5

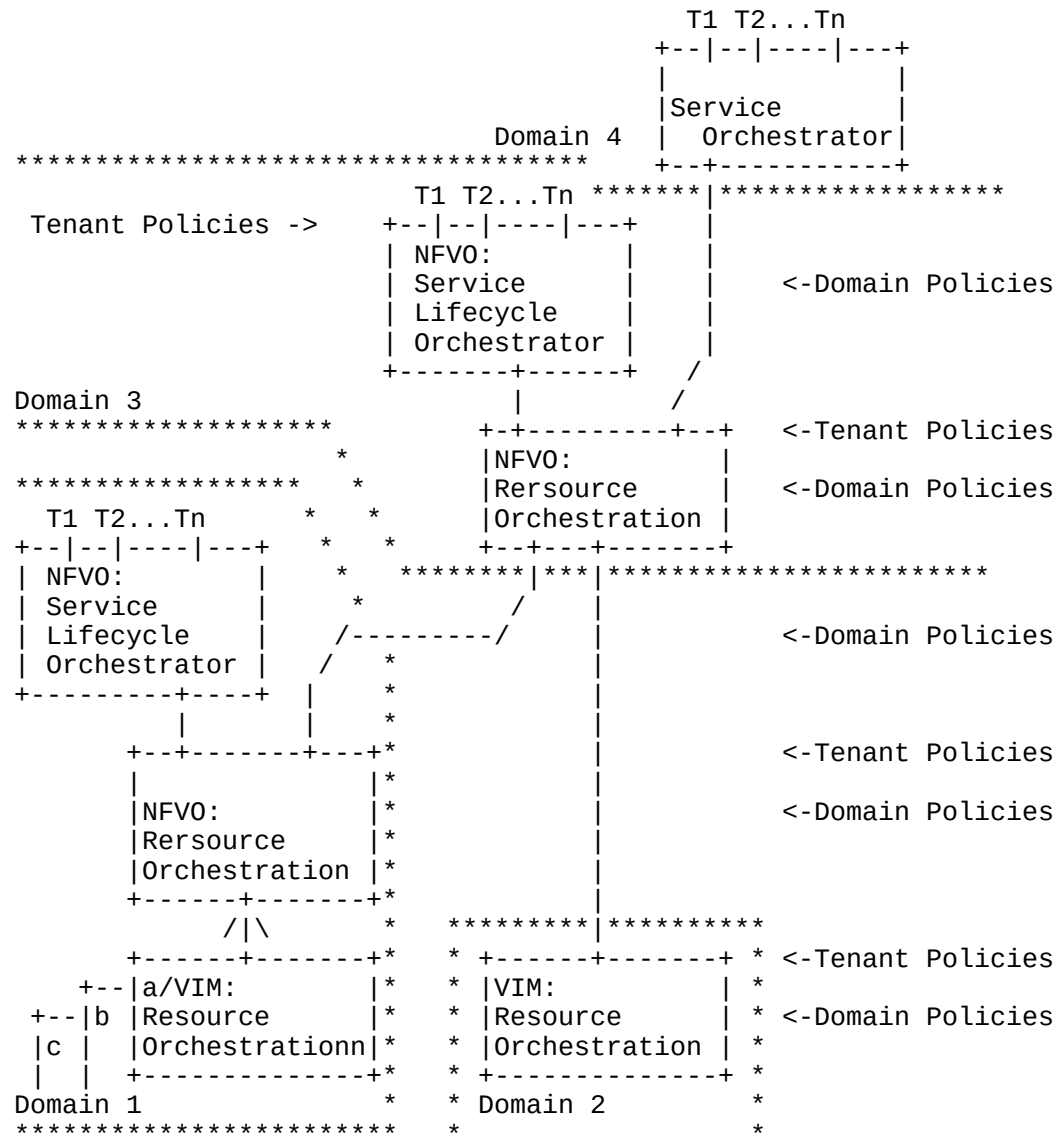


Figure 5: Policies in a Hierarchical Orchestration View

5.5. Policy Pub/Sub Bus

In [I-D.irtf-nfvrg-nfv-policy-arch] the authors argued for the need of policy subsystems to subscribe to policy updates at a higher policy level. A policy publication/subscription (pub/sub) bus would be required as shown in Figure 6.

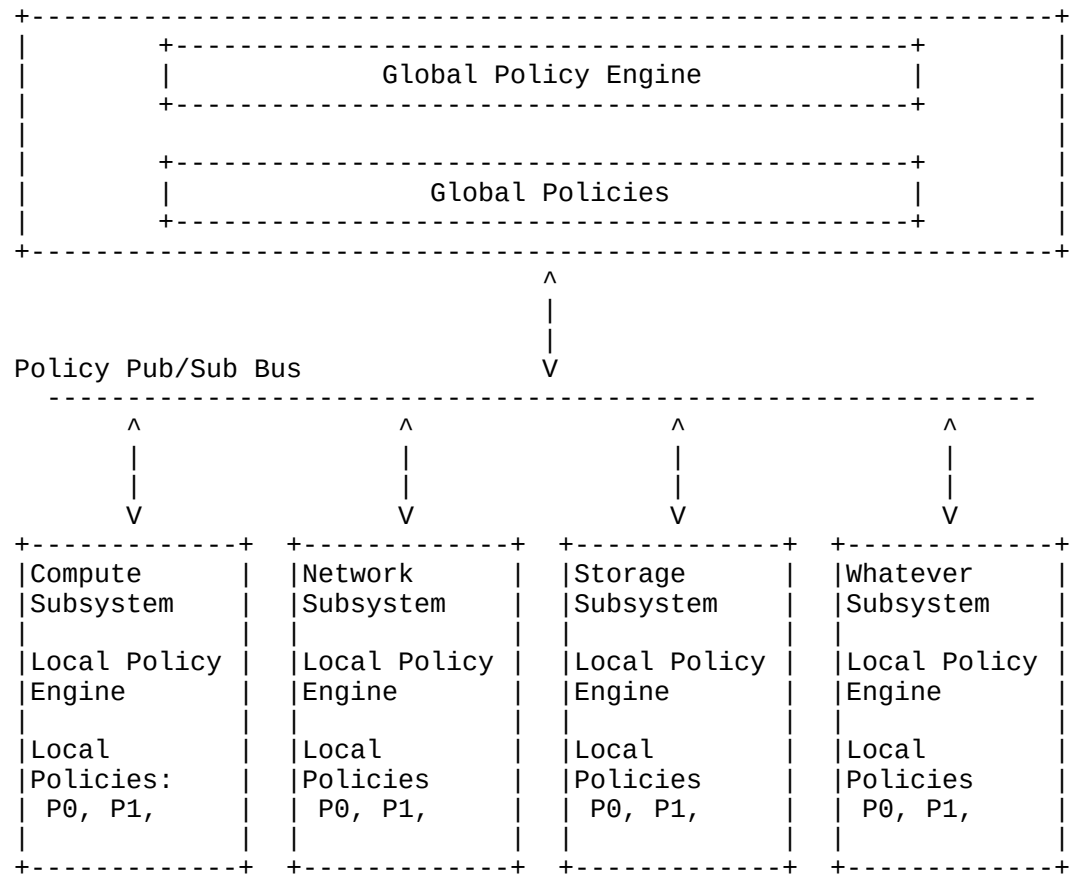


Figure 6: A Policy Pub/Sub Bus

A higher tier policy engine would communicate policies to lower tier policy engines using a policy pub/sub bus. Conversely, lower tier policy engines would communicate their configured policies and services to the higher tier policy engine using the same policy pub/sub bus. Such communications require each policy pub/sub bus to have a pre-defined/pre-configured policy "name space". For example, a pub/sub bus could define services using the name space "Platinum", "Gold", and "Silver". A policy could then be communicated over that pub/sub bus specifying a Silver service requirement.

In a hierarchical policy framework, a policy engine may use more than one policy pub/sub bus, e.g., a policy pub/sub bus named "H" to communicate with a higher tier policy engine and a policy pub/sub bus named "L" to communicate with lower tier policy engines. As the name spaces of policy pub/sub buses H and L may be different, the policy

engine would translate policies defined using the policy pub/sub bus H name space to policies defined using the policy pub/sub bus L name space, and vice-versa.

5.5.1. Pub/sub bus in the hierarchical framework

Figure 7 shows the Pub/sub bus in the hierarchical MANO framework. Policy communications would employ a policy pub/sub bus between the subsystems' policy engines in the policy hierarchy (see Section 5.4). The global NFVO subsystem should have visibility into the policies defined locally at each PoP to be able to detect any potential global policy conflicts, e.g., a local PoP administrator could add a local policy that violates or conflicts with a global policy. In addition, the global NFVO subsystem would benefit from being able to import the currently configured services at each PoP. The global NFVO would use such information to monitor global policy conformance and also to facilitate detection of policy violations when new global policies are created, e.g., a global level administrator is about to add a new global policy that, if committed, would make certain already configured services a violation of the policy. The publication of subsystem service tables for consumption by a global policy engine is a concept used in the Congress [OPENSTACK-CONGRESS] OpenStack [OPENSTACK] project.

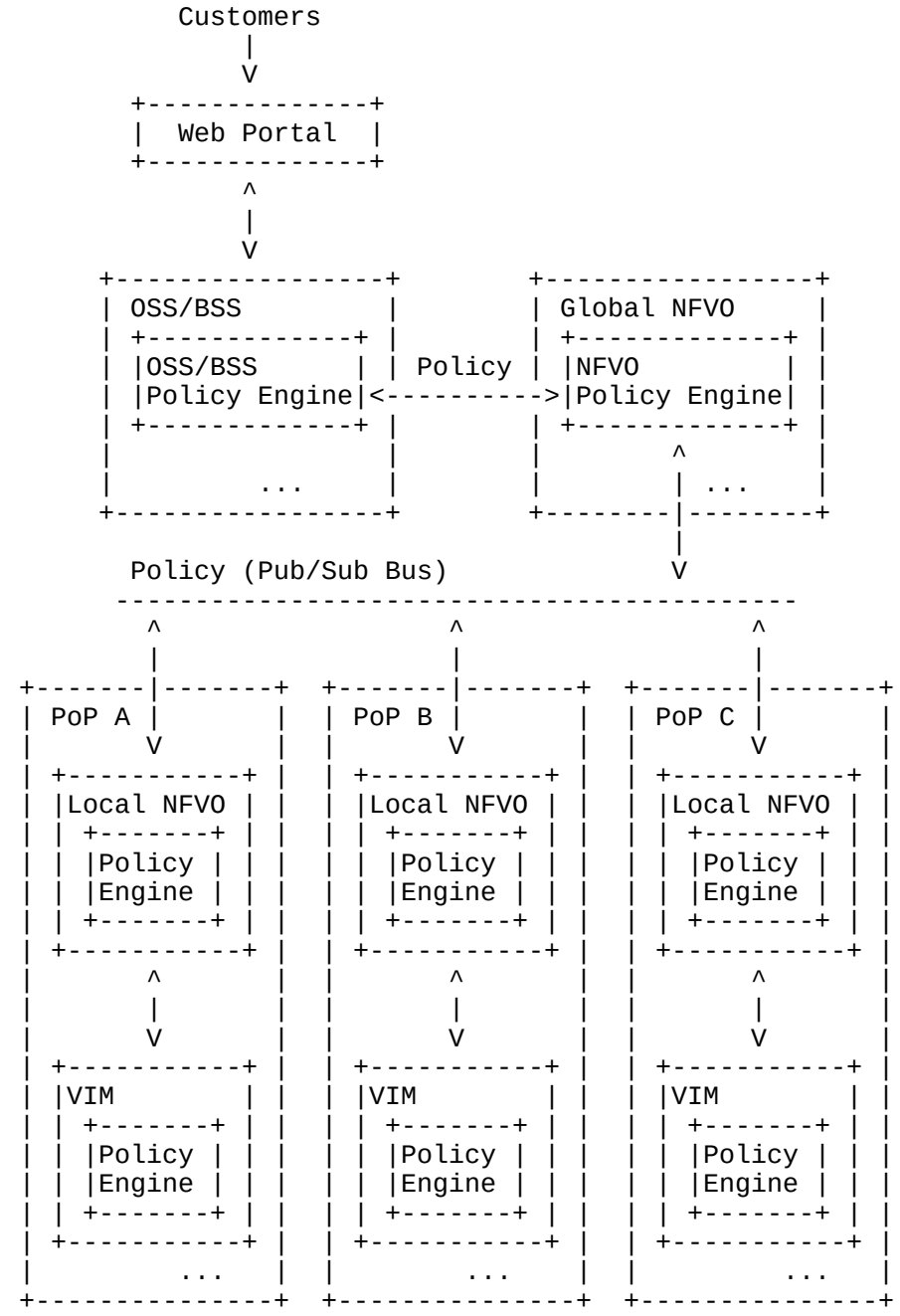


Figure 7: Pub/sub bus in the hierarchical MANO framework

5.6. Policy Intent Statement versus Subsystem Actions and Configurations

Content to be merged

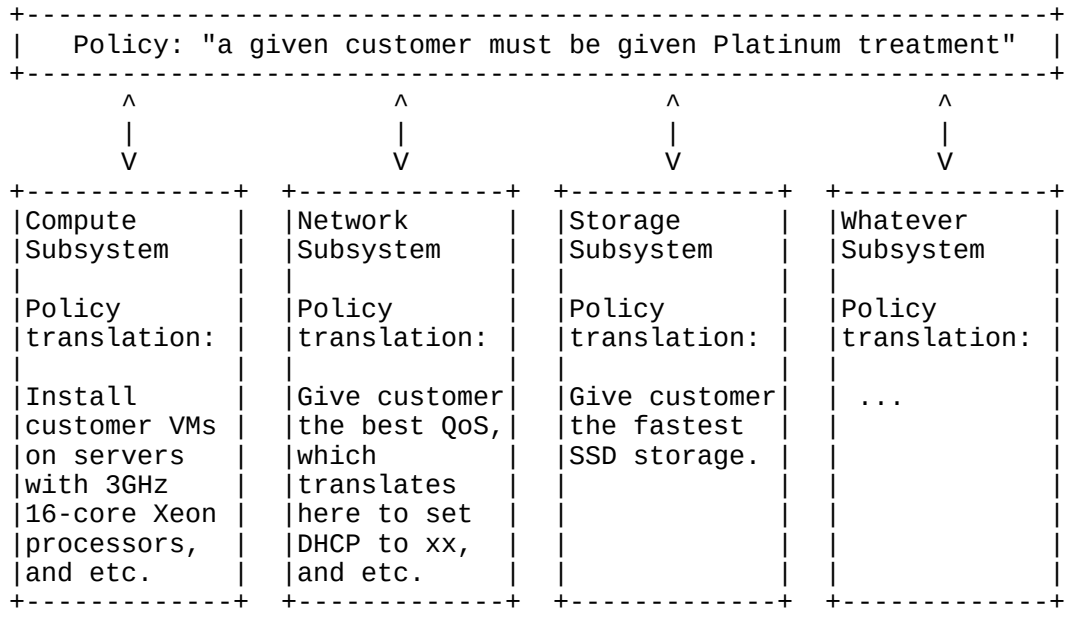


Figure 8: Example of Subsystem Translations of Policy Actions

5.7. Static vs Dynamic vs Autonomic Policies

Content to be merged

5.8. Policy Conflicts and Resolution

Content to be merged

5.9. Soft vs Hard Policy Constraints

Content to be merged

5.10. Operational Policies for Resource management

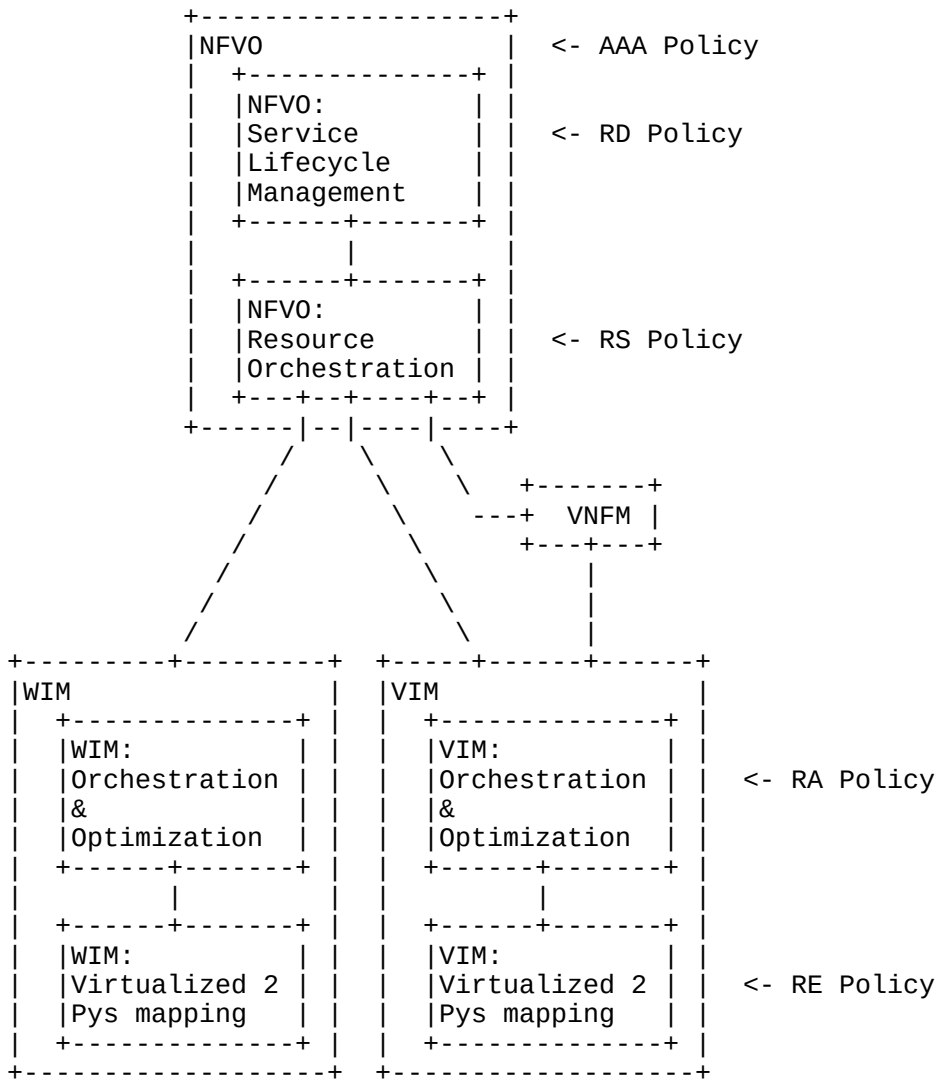


Figure 9: Operational policies for resource management

The use of NFVI resources for multiple network services can be optimized in various objectives as defined in the operational policies (as described in Section 5.2).

The operational policies can be split to different layers of NFVO and VIM/WIM and they include 1) resource scheduling (RS) policy, resource adaptation (RD) policy and authentication, authorization, accounting (AAA) policy at NFVO, and 2) resource allocation (RA) policy and

resource embedding (RE) policy at VIM/WIM. They can be mapped to the MANO architecture as shown in Figure 9.

5.10.1. Operational Policies at NFVO

During NS/VNF lifecycles, states of NFVI/WAN resources or the performance of VNF and VL instances may vary in time (e.g., the performance degradation due to incorrect placement or incorrect forwarding action). Another concern for such dynamic changes is fail-over as a fundamental consideration, i.e., physical resources or virtualized resources in NFVI may fail during network services. These dynamic changes significantly could affect the overall performance for NS. Therefore, such dynamic changes triggered during NS/VNF lifecycles should be coped with for guaranteeing the NS performance and the optimized resource usage. Figure 9 shows that NFVO needs to enforce resource adaptation (RD) policy as an operational policy at NFVO. RD policy supports how NFVO adapts the allocated NFVI/WAN resources (e.g., VM migration, scaling) by dealing with triggered variations. RD policy engine can detect the changes from measurement and diagnosis from VNFM and/or VIM/WIM.

Figure 9 also shows that NFVO needs to enforce resource scheduling (RS) policy. RS policy determines the locations of VNF and VL instances that constitute NS across multiple PoPs and WANs while optimally allocating NFVI and WAN resources to the instances.

In particular, RD and RA policies would consider a business model from OSS/BSS which specifies operational (or business) objectives (e.g., overall energy consumption and NFVI resource utilization) within its domain and with taking account of (on-boarded) network service descriptor (NSD) as an NS policy including the virtualization aspects of application feature, QoS parameters, affinity, anti-affinity rules, and so on.

On the one hand, for the user authorization, authentication, authorization, accounting (AAA) policy may be needed. Authentication policy provides a way of identifying a user while the authorization policy determines whether the user has the authority for virtualized resources (i.e., NFVI/WAN resources) to receive the network service or not. Accounting policy measures the resources the user consumes during the network service. This can include the amount of system time/data, and so on.

5.10.2. Operational Policies at VIM/WIM

As shown in Figure 9, RA policy supports how each subsystem (e.g., compute, storage subsystem) in NFVI is allocated depending on the placement information from NFVO to further optimize the resource

usage. Moreover, the assigned NFVI resources are embedded (or allocated) to physical resources in VIM/WIM depending on states and usage of resources by means of resource embedding (RE) policy as shown in Figure 9. In other words, RE policy determines and coordinates how the allocated virtual resources are mapped to physical resources. For example, RE policy may be updated when some physical resources are failed or a virtualization technique is changed.

6. Policy-Based Resource Management Examples

6.1. Policy-Based Multipoint Ethernet Service

Content to be merged

6.2. Policy-Based NFV Placement

Content to be merged

6.3. Policy-Based VNF-FG Management

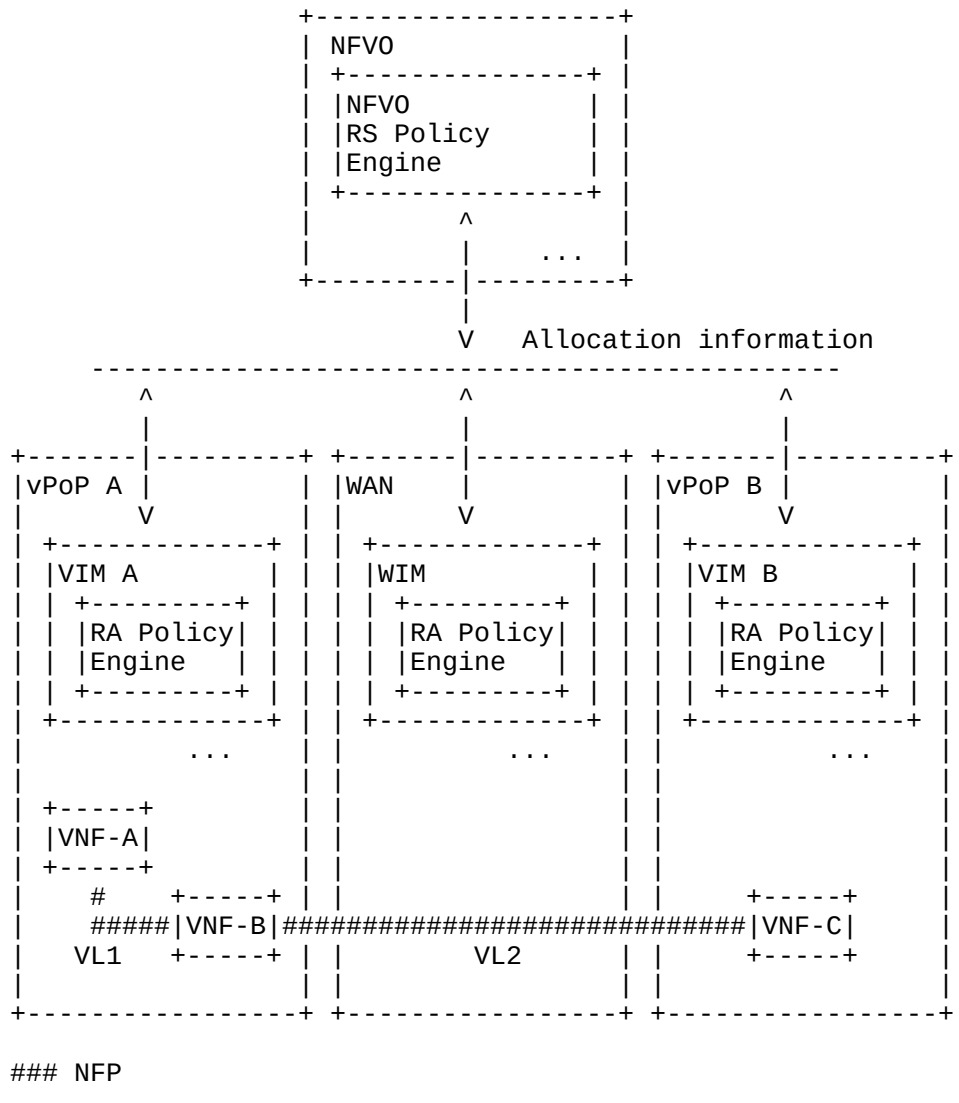


Figure 10: Policy-based VNF-FG Management

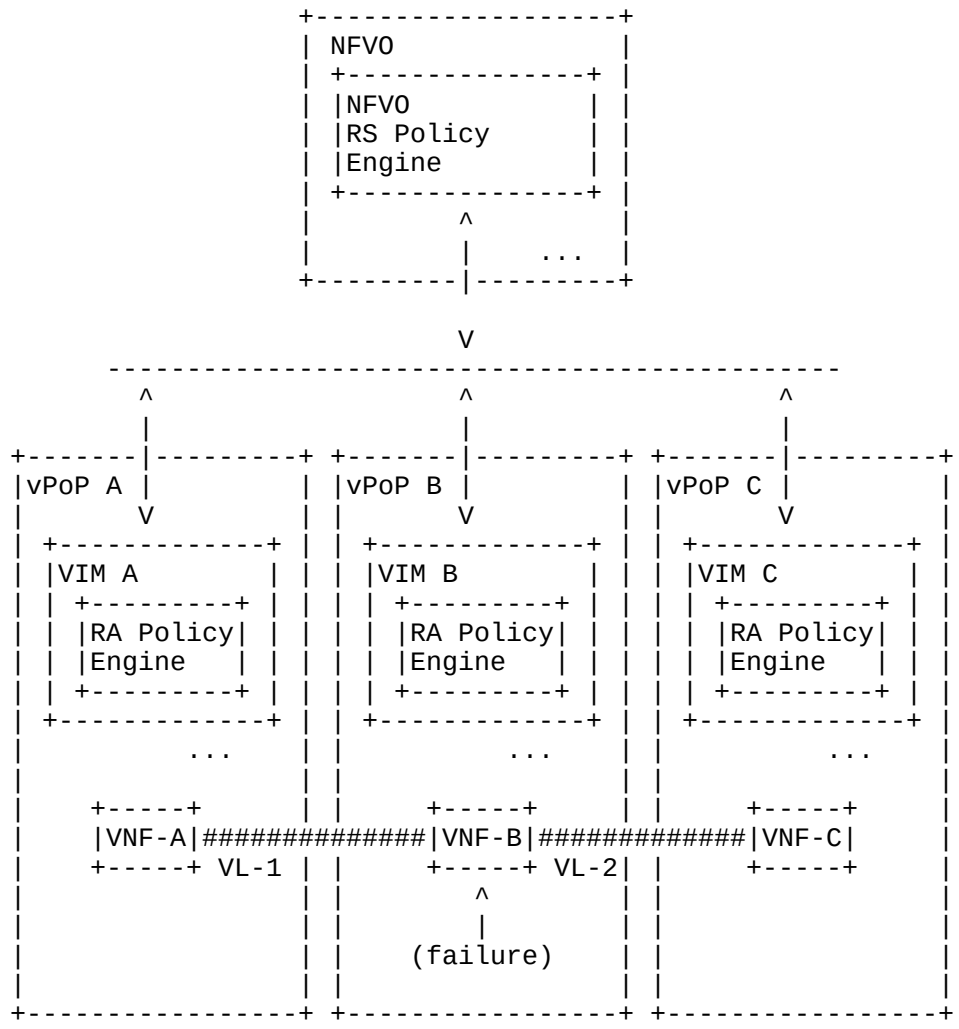
Another subsystem example for the policy framework is VNF-FG. When VNF-FGs of end-to-end network services are realized, NFVI resources across multiple NFVI-PoPs and WAN resources that connect among them should be allocated to the VNF-FGs. It depends on the target KPIs of individual VNF and VL instances that constitute VNF-FGs. In particular, in case of VNF-FG, chained performances and capabilities

of VNF and VL instances need to be considered together with on VL instances the inter-connectivity between different NFVI-PoPs. For example, if one of the VNF instances or VL instances along the VNF-FG gets overloaded, the end-to-end network service may also get affected. Therefore, while features of such VNF-FG are carefully considered, proper operational policies for resource management (see Section 5.10) are required.

As shown in Figure 10, consider a scenario where a user requests a VNF-FG composed of "VNF A-VL 1-VNF B-VL 2-VNF C". For the VNF-FG, an RA policy is enforced in which it is designed to avoid over-utilization of PoP A and to reduce latency on VL 1. Therefore, NFVO places VNF A, VNF B, and VL 1 on PoP A by consuming its computing and network resources to achieve low latency. On the other hand, VL 2 and VNF C is allocated to the resources of WAN and PoP B, respectively to avoid over-utilization of PoP A.

On the one hand, dynamic changes such as a VNF failure significantly affect on the overall performance of VNF-FG since VNF-FG is a chain of VNF and VL instances. Thus, such dynamic changes should be coped with by RD policy for guaranteeing the VNF-FG performance and the optimized resource usage. A fault management for VNF-FG based on policy example is shown in Section 6.4.

6.4. Policy-Based Fault Management



NFP

Figure 11: Failure Scenario for VNF-FG

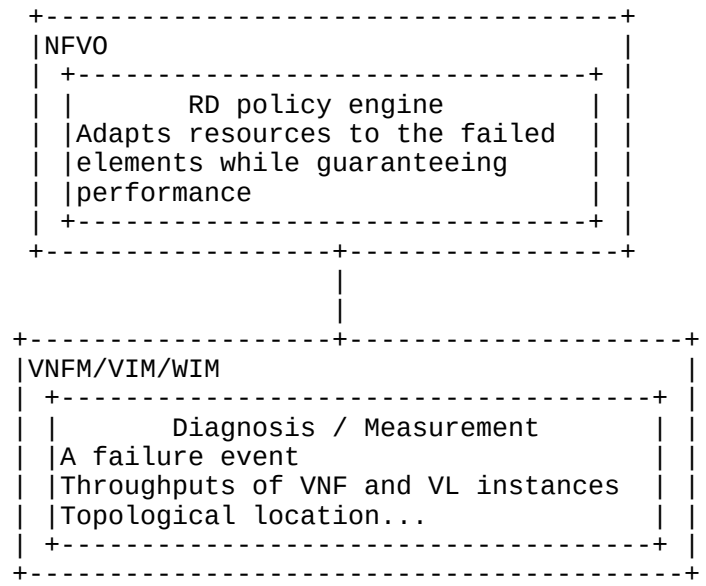


Figure 12: Architecture for policy-based fault management

As shown in Figure 11, consider a scenario that a VM related to VNF-B (i.e., a VNF-B instance) is failed in the given VNF-FG composed VNF-A, VNF-B, VNF-C in order. Note that the NFVI and WAN resources are already allocated to the instances by RS policy. For service continuity, failure of the VNF-B instance needs to be detected based on diagnosis function in VIM/VNFM and the failed one needs to be replaced with a new instance or to be assigned to the existing instance which is available. The diagnosis and measurement function may collect current performance measures and location for instances as well as such a failure event.

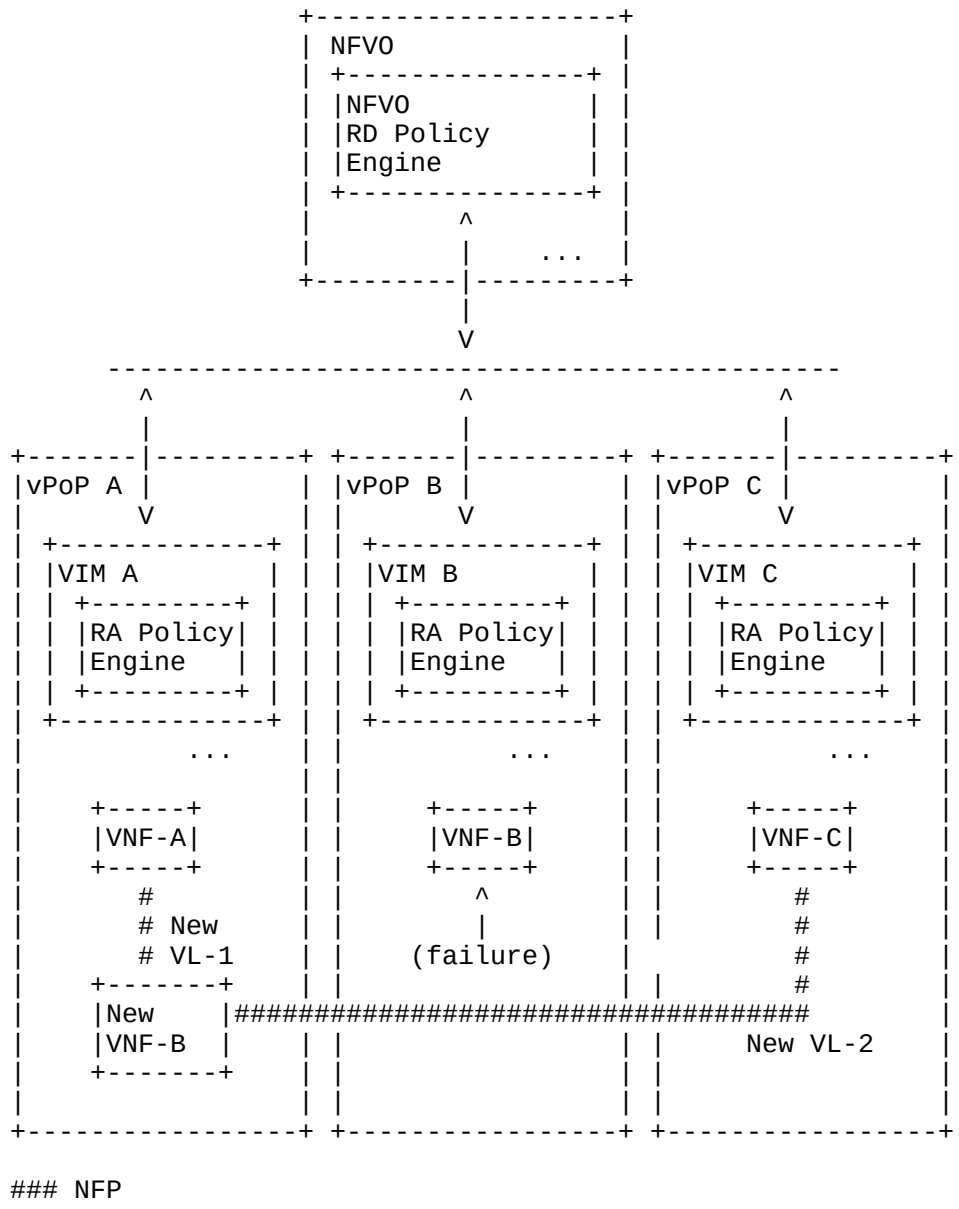


Figure 13: Re-instantiation for VNF-FG

In the first case where a VNF instantiation is needed, a new VNF instantiation is determined by the RD policy engine in NFVO. For

example, NFVO may avoid replacement of VNF B on NFVI-PoP B owing to high possibility of failure. Therefore, NFVO could instantiate VNF B on NFVI-PoP A or NFVI-PoP C with the setup of new connection points (CPs) while guaranteeing performance as shown in Figure 13.

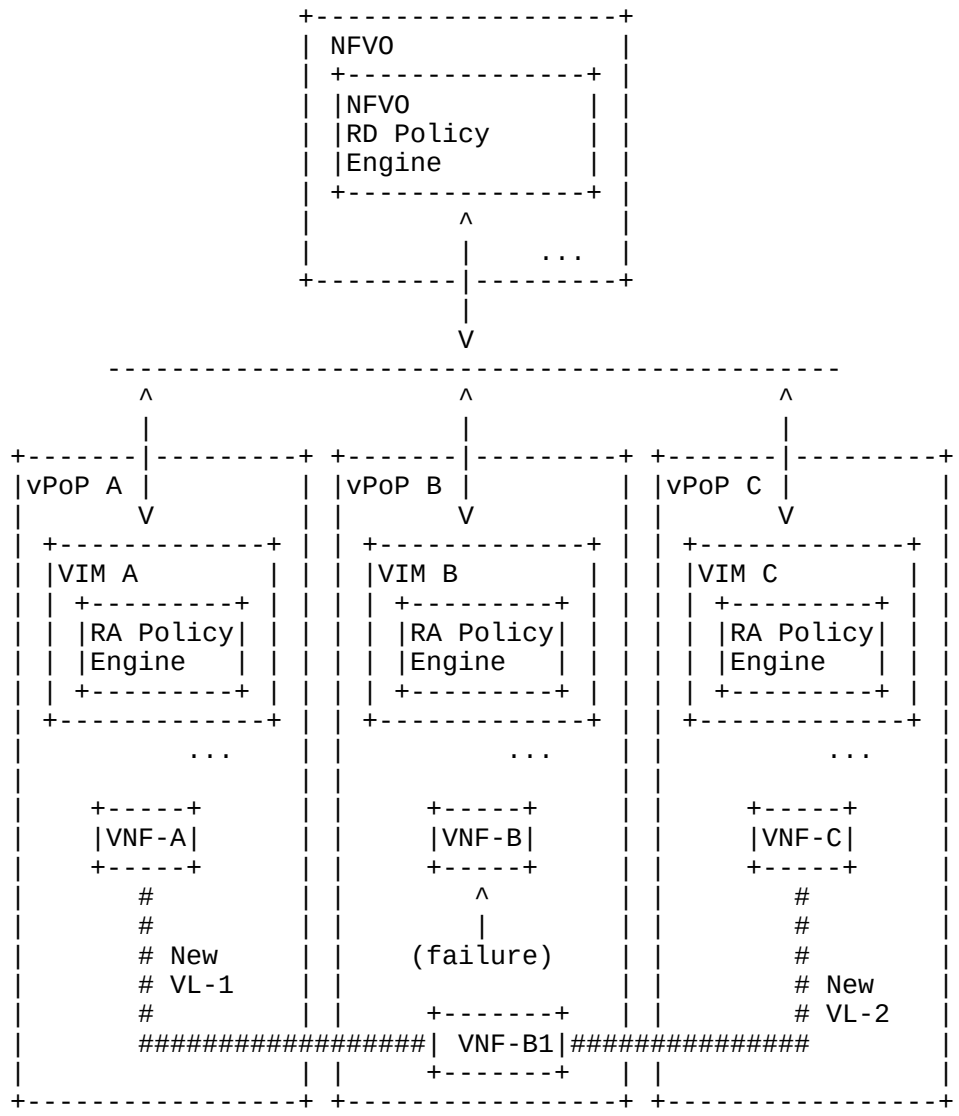


Figure 14: No Re-instantiation for VNF-FG

In the second case where no VNF instantiation is needed since a redundant VNF exists, the available VNF-B instance can used by the VNF-FG. For example, a redundant VNF B instance exists in NFVI-PoP

B. Therefore, NFVO selects the instance and re-constructs two VLs as shown in Figure 14, and the corresponding NS can be continued without re-instantiation.

7. Implementation Examples

tbd

8. Gaps and Open Questions

tbd

9. Conclusions

tbd

9.1. Relation to other IETF/IRTF activities

tbd

10. Acknowledgements

The research leading to some of the results described in this document has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 619609 - the UNIFY project. 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. Contributors

This document is the result of merging multiple drafts. This section acknowledges those who provided important ideas and text into this document.

- Z. Qiang (Ericsson), M. Kind (DT-AG) from [I-D.unify-nfvrg-recursive-programming]
- R. Krishnan (Dell), D. Lopez (Telefonica) and S. Wright (AT&T) from [I-D.irtf-nfvrg-nfv-policy-arch]
- S. Lee (ETRI), S. Pack (KU), M-K. Shin (ETRI) and E. Paik (KT) from [I-D.irtf-nfvrg-resource-management-service-chain]

12. IANA Considerations

tbd

13. Security Considerations

tbd

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3060] Moore, B., Ellesson, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, DOI 10.17487/RFC3060, February 2001, <http://www.rfc-editor.org/info/rfc3060>.

[RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, DOI 10.17487/RFC3198, November 2001, <http://www.rfc-editor.org/info/rfc3198>.

[RFC3670] Moore, B., Durham, D., Strassner, J., Westerinen, A., and W. Weiss, "Information Model for Describing Network Device QoS Datapath Mechanisms", RFC 3670, DOI 10.17487/RFC3670, January 2004, <http://www.rfc-editor.org/info/rfc3670>.

14.2. Informative References

[CERI-DATALOG] Ceri, S. and others, "What you always wanted to know about Datalog (and never dared to ask", IEEE Transactions on Knowledge and Data Engineering, (Volume: 1, Issue: 1), August 2002.

[ETSI-NFV-MANO] ETSI, "Network Function Virtualization (NFV) Management and Orchestration V0.6.3", October 2014.

[ETSI-NFV-PER001] ETSI, "Network Function Virtualization: Performance and Portability Best Practices v1.1.1", June 2014.

[ETSI-NFV-TERM]
ETSI, "NFV Terminology for Main Concepts in NFV", October 2013.

[ETSI-NFV-WHITE-PAPER]
ETSI NFV White Paper, "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges, & Call for Action", <http://portal.etsi.org/NFV/NFV_White_Paper.pdf>.

[I-D.ietf-bmwg-virtual-net]
Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", draft-ietf-bmwg-virtual-net-04 (work in progress), August 2016.

[I-D.irtf-nfvrg-nfv-policy-arch]
Figueira, N., Krishnan, R., Lopez, D., Wright, S., and D. Krishnaswamy, "Policy Architecture and Framework for NFV Infrastructures", draft-irtf-nfvrg-nfv-policy-arch-04 (work in progress), September 2016.

[I-D.irtf-nfvrg-resource-management-service-chain]
Lee, S., Pack, S., Shin, M., Paik, E., and R. Browne, "Resource Management in Service Chaining", draft-irtf-nfvrg-resource-management-service-chain-03 (work in progress), March 2016.

[I-D.liu-bmwg-virtual-network-benchmark]
Liu, V., Liu, D., Mandeville, B., Hickman, B., and G. Zhang, "Benchmarking Methodology for Virtualization Network Performance", draft-liu-bmwg-virtual-network-benchmark-00 (work in progress), July 2014.

[I-D.norival-nfvrg-nfv-policy-arch]
Figueira, N., Krishnan, R., Lopez, D., and S. Wright, "Policy Architecture and Framework for NFV Infrastructures", draft-norival-nfvrg-nfv-policy-arch-04 (work in progress), June 2015.

[I-D.unify-nfvrg-recursive-programming]
Szabo, R., Qiang, Z., and M. Kind, "Towards recursive virtualization and programming for network and cloud resources", draft-unify-nfvrg-recursive-programming-02 (work in progress), October 2015.

[ODL-GB-POLICY]
"OpenDaylight Group Based Policy", <https://wiki.opendaylight.org/view/Project_Proposals:Group_Based_Policy_Plugin>.

[ODL-NIC-PROJECT]
"OpenDaylight Network Intent Composition Project", <https://wiki.opendaylight.org/index.php?title=Network_Intent_Composition:Main#Friday_8AM_Pacific_Time>.

[ODL-SDN-CONTROLLER]
"OpenDaylight SDN Controller", <http://www.opendaylight.org/>.

[OPENSTACK]
"OpenStack", <http://www.openstack.org/>.

[OPENSTACK-CONGRESS]
"OpenStack Congress", <https://wiki.openstack.org/wiki/Congress>.

[OPENSTACK-NEAT]
"OpenStack Neat", <http://openstack-neat.org/>.

[OPENSTACK-NEUTRON]
"OpenStack Neutron", <https://wiki.openstack.org/wiki/Neutron>.

[POLICY-FRAMEWORK-WG]
"Policy Framework Working Group (IETF)", <http://www.ietf.org/wg/concluded/policy.html>.

[RESOURCE-MGMT-SERVICE-CHAIN]
Lee, S. and others, "Resource Management in Service Chaining", <https://datatracker.ietf.org/doc/draft-irtf-nfvrg-resource-management-service-chain/>.

[SDN-MULTI-DOMAIN]
Figueira, N. and R. Krishnan, "SDN Multi-Domain Orchestration and Control: Challenges and Innovative Future Directions", IEEE International Conference on Computing (ICNC), February 2015.

[VM-HOSTING-NET-CLUSTER]
Grit, L. and others, "Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration", Virtualization Technology in Distributed Computing (VTDC), 2006.

Authors' Addresses

Robert Szabo (editor)
Ericsson
Konyves Kaman krt. 11
Budapest, EMEA 1097
Hungary

Phone: +36703135738
Email: robert.szabo@ericsson.com

Seungik Lee (editor)
ETRI
218 Gajeong-ro Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Norival Figueira
Brocade

Email: nfigueir@Brocade.com

NFV Research Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2017

M-K. Shin, Ed.
ETRI
K. Nam
Friesty
S. Pack
KU
S. Lee
ETRI
R. Krishnan
Dell
March 6, 2017

Verification of NFV Services : Problem Statement and Challenges
draft-irtf-nfvrg-service-verification-03

Abstract

NFV relocates network functions from dedicated hardware appliances to generic servers, so they can run in software. However, incomplete or inconsistent configuration of virtualized network functions (VNFs) and forwarding graph (FG, aka service chain) could cause break-down of the supporting infrastructure. In this sense, verification is critical for network operators to check their requirements and network properties are correctly enforced in the supporting infrastructures. Recognizing these problems, we discuss key properties to be checked on NFV services. Also, we present challenging issues related to verification in NFV environments.

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 September 6, 2017.

Copyright Notice

Table of Contents

1. Introduction

NFV is a network architecture concept that proposes using IT virtualization related technologies, to virtualize entire classes of network service functions into building blocks that may be connected, or chained, together to create network services. NFV service is defined as a composition of network functions and described by its functional and behavioral specification, where network functions (i.e., firewall, DPI, SSL, load balancer, NAT, AAA, etc.) are well-defined, hence both their functional behavior as well as their external interfaces are described in each specifications.

In NFV, a VNF is a software package that implements such network functions. A VNF can be decomposed into smaller functional modules or APIs for scalability, reusability, and/or faster response [ETSI-NFV-Arch],[ETSI-NFV-MANO]. This modular updates or composition for a network function may lead to many other verification or security issues. In addition, a set of ordered network functions which build FGs may be connected, or chained, together to create an end-to-end network service. Multiple of VNFs can be composed together to reduce management and VNF FGs. While autonomic networking techniques could be used to automate the configuration process including FG updates, it is important to take into account that incomplete and/or inconsistent configuration may lead to verification issues. Moreover, automation of NFV process with integration of SDN may lead the network services to be more error-prone. In this sense, we need to identify and verify key properties to be correct before VNFs and FGs are physically placed and realized in the supporting infrastructure.

1.1. Terminology

This document draws freely on the terminology defined in [ETSI-NFV-Arch].

2. Problem statement

The verification services should be able to check the following properties:

2.1. Dependencies of network service components in NFV framework

In NFV framework, there exist several network service components including NFVI, VNFs, MANO, etc. as well as network controller and switches to realize end-to-end network services. Unfortunately, these components have intricate dependencies that make operation incorrect. In this case, there is inconsistency between states stored and managed in VNF FGs and network tables (e.g., flow tables), due to

communication delays and/or configuration errors. For example, if a VNF is replicated into the other same one for the purpose of load balance and a new FG is established through the copied one, but all the state/DBs replication is not finished yet due to delays, this can be lead to unexpected behaviors or errors of the network service. Therefore, these dependencies make it difficult to correctly compose NFV-enabled end-to-end network services.

2.2. Invariant and error check in VNF FGs

In VNF FGs, an infinite loop construction should be avoided and verified. Let us consider the example. Two VNF A and VNF B locate in the same service node X whereas another VNF C resides in other service node Y [SIGCOMM-Gember]. Also, the flow direction is from X toY, and the given forwarding rule is A->C->B. In such a case, service node Y can receive two ambiguous flows from VNF A: 1) one flow processed by VNF A and 2) another flow processed by VNF A, B, and C. For the former case, the flow should be processed by VNF C whereas the latter flow should be further routed to next service nodes. If these two flows cannot be distinguished, service node Y can forward the flow to service node X even for the latter case and a loop can be formed. To avoid the infinite loop formation, the forwarding path over VNF FG should be checked in advance with the consideration of physical placement of VNF among service nodes. Also, reactive verification may be necessary, since infinite loop formation may not be preventable in cases where configuration change is happening with live traffic.

In addition, isolation between VNFs (e.g. confliction of properties or interference between VNFs) and consistent ordering of VNF FGs should be always checked and maintained.

2.3. Load balancing among VNF instances

In VNF FG, different number of VNF instances can be activated on several service nodes to carry out the given task. In such a situation, load balancing among the VNF instances is one of the most important considerations. In particular, the status in resource usage of each service node can be different and thus appropriate amount of jobs should be distributed to the VNF instances. To guarantee well-balanced load among VNF instances, the correctness of hash functions for load balancing needs to be verified. Moreover, when VNF instances locate in physically different service nodes, simple verification of load balancing in terms of resource usage is not sufficient because different service nodes experience diverse network conditions (e.g., different levels of network congestion)[ONS- Gember]. Therefore, it is needed to monitor global network condition as well as local resource condition to achieve the network-wide load balancing in VNF

FGs. Also, whether the monitoring function for network/compute/storage resources is correctly working should be checked.

2.4. Policy and state consistency on NFV services

In VNF FG, policy to specific users can be dynamically changed. For example, a DPI VNF can be applied only in the daytime in order to prohibit from watching adult contents while no DPI VNFs applied during the nighttime. When the policy is changed, the changed policy should be reconfigured in VNF service nodes as soon as possible. If the reconfiguration procedure is delayed, inconsistent policies may exist in service nodes. Consequently, policy inconsistency or confliction needs to be checked. Also in some situations, states for VNF instances may be conflicted or inconsistent. Especially when a new VNF instance is instantiated for scale-up and multiple VNF instances are running, these multiple VNF instances may have inconsistent states owing to inappropriate instantiation procedure [SIGCOMM-Gember]. In particular, since the internal states of VNF instances (e.g., the instantaneous state of CPU, register, and memory in virtual machine) are not easily-visible, a new way to check the VNF internal states should be devised.

2.5. Performance

In VNF FG, VNF instances can locate in different service nodes and these service nodes have different load status and network conditions. Consequently, the overall throughput of VNF FG is severely affected by the service nodes running VNF instances. For example, if a VNF instance locates in a heavily loaded service node, the service time at the service node will be increased. In addition, when a VNF FG includes a bottleneck link with network congestion, the end-to-end performance (e.g., latency and throughput) in the VNF FG can be degraded. Therefore, the identification of bottleneck link and node is the first step for performance verification or guarantee of the VNF FG [ONS-Gember]. After detecting the bottleneck link/node, the VNF requiring scale up or down can be identified and the relocation of VNF instance among service nodes can be determined

2.6. Security

How to verify security holes in VNF FG is another important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, several VNFs (e.g., NAT) can modify or update packet headers and payload. In these environments, it is difficult to protect the integrity of flows traversing such VNFs. Another security concern in the VNF FG is distributed denial of service (DDoS) to a

specific service node. If an attacker floods packets to a target service node, the target service node cannot perform its functions correctly. Therefore, such security attacks in the VNF FG should be detected and handled in an efficient manner. In the case of DDoS, adding a DDoS appliance as the first element in the service chain would help alleviate the problem. Moreover, unknown or unauthorized VNFs can run and thus how to identify those problems is another security challenge.

3. Examples - NS policy conflict with NFVI policy

Another target of NFV verification is conflict of NS policies against global network policy, called NFVI policy.

NFV allocates and manages NFVI resources for a network service according to an NS policy given in the network service descriptor (NSD), which describes how to govern NFVI resources for VNF instances and VL instances to support KPIs of the network service. Example factors of the NS policy are resource constraints (or deployment flavor), affinity/anti-affinity, scaling, fault and performance management, NS topology, etc.

For a network-wide (or NS-wide) management of NFVI, NFVI policy (or global network policy) can be provided to describe how to govern the NFVI resources for optimized use of the infrastructure resources (e.g., energy efficiency and load balancing) rather than optimized performance of a single network service. Example factors of the NFVI policy are NFVI resource access control, reservation and/or allocation policies, placement optimization based on affinity and/or anti-affinity rules, geography and/or regulatory rules, resource usage, etc.

While both of the policies define the requirements for resource allocation, scheduling, and management, the NS policy is about a single network service; and the NFVI policy is about the shared NFVI resources, which may affect all of the given network services globally. Thus, some of NS and NFVI policies may be inconsistency with each other when they have contradictive resource constraints on the shared NFVI resources. Examples of the policy conflicts are as follows:

<Example conflict case #1>

- NS policy of NS_A (composed of VNF_A and VNF_B)
  - Resource constraints: 3 CPU core for VNF_A and 2 CPU core for VNF_B
  - Affinity rule between VNF_A and VNF_B

o NFVI policy
- No more than 4 CPU cores per physical host

o Conflict case
- The NS policy cannot be met within the NFVI policy

<Example conflict case #2>

o NS policy of NS_B (composed of VNF_A and VNF_B)
- Affinity rule between VNF_A and VNF_B

o NFVI policy
- Place VM whose outbound traffic is larger than 100Mbps at POP_A
- Place VM whose outbound traffic is smaller than 100Mbps at POP_B

o Conflict case
- If VNF_A and VNF_B generate traffic in 150Mbps and 50Mbps, respectively,
- VNF_A and VNF_B need to be placed at POP_A and POP_B, respectively according to the NFVI policy
- But it will violate the affinity rule given in the NS policy

<Example conflict case #3>

o NS policy of NS_C (composed of VNF_A and VNF_B)
- Resource constraints: VNF_A and VNF_B exist in the same POP
- Auto-scaling policy: if VNF_A has more than 300K CPS, scale-out

o NFVI policy
- No more than 10 VMs per physical host in POP_A

o Conflict case
- If CPS of VNF_A in POP_A gets more than 300K CPS,
- and if there is no such physical host in the POP_A whose VMs are smaller than 10,
- VNF_A need to be scaled-out to other POP than POP_A according to the NFVI policy
- But it will violate the NS policy

4. Requirements of verification framework

The verification framework addressed in this document follows [ETSI-NFV-Testing]. [ETSI-NFV-Testing] covers the following aspects of pre-deployment testing: 1) assessing the performance of the NFVI and its ability to fulfil the performance and reliability requirements of the VNFs executing on the NFVI, 2) data and control plane testing of VNFs and their interactions with the NFV Infrastructure and the NFV MANO,

and 3) validating the performance, reliability and scaling capabilities of network services.

A verification framework for NFV-based services also needs to satisfy the following requirements:.

- o R1 : It should be able to check global and local properties and invariants. Global properties and invariants relate to the entire VNFs, and local properties and invariants relates to the specific domain or resources that some of the VNFs are using. For example, Loop-freeness and isolation between VNFs can be regarded as global. The policies that are related only to the specific network controllers or devices are local.

- o R2 : It should be able to access to the entire network states whenever verification tasks are started. It can directly manage the states of network and NFV-based services through databases or any solution that specializes in dealing with the network topology and configurations, or can utilize the functions provided by NFV M&O and VNFI solutions to get or set the states at any time.

- o R3 : It should be independent from specific solutions and frameworks, and provide standard APIs.

- o R4 : It should process standard protocols such as NetConf, YANG, OpenFlow, and northbound and southbound interfaces that are related network configurations, and used by OSS.

5. Challenging issues

There are emerging challenges that the verification services face with.

5.1. Consistency check in distributed state

Basically, NFV states as well as SDN controllers are distributed. writing code that works correctly in a distributed setting is very hard. Therefore, distributed state management and consistency check has challenging issues. Some open source project such as ONOS offers a core set of primitives to manage this complexity. RAFT algorithm [RAFT] is used for distribution and replication. Similarly, Open daylight project has a clustering concept to management distributed state. There is no "one-size-fits-all" solution for control plane data consistency.

5.2. Intent-based service composition

Recently, Intent-based high-level language is newly proposed and discussed in open source project. The Intent allows for a descriptive way to get what is desired from the infrastructure, unlike the current NFV description and SDN interfaces which are based on describing how to provide different services. This Intent will accommodate orchestration services and network and business oriented SDN/NFV applications, including OpenStack Neutron, Service Function Chaining, and Group Based Policy. A Intent compiler that translates and compiles it into low level instructions (e.g., SDN controller/OpenStack primitives) for network service components. In this sense, error checking and debugging are critical for reliable Intent-based service composition.

5.3. Finding infinite loops

General solutions for the infinite loop can lead to intractable problem (e.g. the halting problem). To make the verification practical and minimize the complexity, some of the restrictions are required. Finding cycle can be processed in polynomial time but the restriction could be too much for some cases that service functions or network flows requires finite loops.

5.4. Live traffic verification

It is known fact that the complexity of verification tasks for the real and big problem is high. A few invariants can be checked in real-time but it would be impossible if the size of VNFs increases or properties to be checked are complex.

5.5. Languages and their semantics

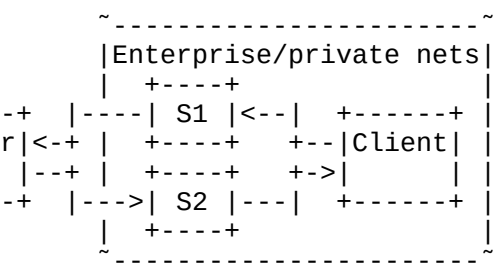
For the verification, configurations and states of VNFs need to be precisely expressed using formal semantics. There are many languages and models, and it is impractical for the verification frameworks to support all of the existing languages and models. Languages and semantic models optimized to the verification framework need to selected or newly developed.

5.6. Stateful VNFs with multiple physical views

The correctness of VNFs whose behaviors depend on the previous states (packets, actions, etc) and whose physical entities are multiple should be checked differently than the stateless ones. Such VNFs include firewall, load balancer, NAT, flow rules with counter or soft timeout.

o Case 1:
If a firewall service is implemented over two physical Openflow switches, there could be two paths that the client-server packets go through. If the packets between client and server go through the same switch, firewall functions correctly. However if packets from client to server go through S1 but packets from server to client come back through S2, those flows could be blocked and lead to false-negative result.

To mitigate the situation, states of all instances for one logical VNF must be considered to verify the correctness.



o Case 2:
If there are VNFs whose behavior depend on the previous VNF, those dependency must be considered as well.

For example, if firewall and load balancer gets packets go through NAT service, they need to know the header mapping information that the NAT have set to correctly process their functions. If the FG consists of IPS followed by DPI and those functions are connected different switches, the switch connecting DPI must know if the incomming packets should be forwarded to DPI or not. Port knocking is also well-known example of stateful function.

To mitigate the situation, the states of all VNFs having behavioral dependency must be considered when they are verified.

6. Gap analysis - open source projects

Recently, the Open Platform for NFV (OPNFV) community is collaborating on a carrier-grade, integrated, open source platform to accelerate the introduction of new NFV products and services [OPNFV]. Open Daylight (ODL) is also being tightly coupled with this OPNFV platform to integrate SDN controller into NFV framework [ODL].

This clause analyzes the existing open source projects including

OPNFV and ODL related to verification of NFV services.

6.1. OPNFV

6.1.1. Doctor

The Doctor project provides a NFVI fault management and maintenance framework on top of the virtualized infrastructure. The key feature is to notify unavailability of virtualized resources and to recover unavailable VNFs.

While the Doctor project focuses only on faults in NFVI including compute, network, and storage resources, the document discusses broader fault management issues such as break-down of the supporting infrastructure due to incomplete or inconsistent configuration of NFV services.

6.1.2. Prediction

The Prediction project provides a data collection for failure prediction framework. The failure prediction framework diagnoses or verifies which entity is suspected to be progressing towards a failure and which VNFs might be affected due to the predicted anomaly.

While the Prediction project focuses only on fault prediction in NFVI compute, network, and storage resources, the document includes broader fault management and prediction issues such as faults in the NFV service deployment and operation.

6.1.3. Resource Scheduler

The Resource Scheduler project provides an enhanced scheduler for optimizing the performance of the VNFs. In particular, this project supports resource isolation. For example, when a VNF strictly requires low latency, strongly isolated compute resources can be allocated to the VNF.

The Resource Scheduler project only focuses on optimizing the performance of individual VNFs without considering the end-to-end performance (e.g., latency and throughput) in NFV services.

6.1.4. Moon

The Moon project implements a security management system for the cloud computing infrastructure. The project also enforces the security managers through various mechanisms, e.g., authorization for access control, firewall for networking, isolation for storage, and

logging for tractability.

Note that the main interest of the Moon project is the DDoS attack to a service node and the IDS management for VNFs. A wider range of security issues in the NFV service verification need to be discussed.

6.1.5 Bottlenecks

The Bottlenecks project aims to find system bottlenecks by testing and verifying OPNFV infrastructure in a staging environment before committing it to a production environment. Instead of debugging the deployment in production environment, an automatic method for executing benchmarks to validate the deployment during staging is adopted. For example, the system measures the performance of each VNF by generating workload on VNFs.

The Bottlenecks project does not consider incomplete or inconsistent configurations on NFV services that might cause the system bottlenecks. Furthermore, the Bottlenecks project aims to find system bottlenecks before committing it to a production environment. Meanwhile, the draft also considers how to find bottlenecks in real time.

6.2. ODL

6.2.1. Network Intent Composition

The Network Intent Composition project enables the controller to manage and direct network services and network resources based on intent for network behaviors and network policies. Intents are described to the controller through a new northbound interface, which provides generalized and abstracted policy semantics. Also, the Network Intent Composition project aims to provide advanced composition logic for identifying and resolving intent conflicts across the network applications.

When the reconfiguration upon the policy (i.e, intent) is delayed, policy inconsistency in service nodes may occur after the policy is applied to service nodes. While the Network Intent Composition project resolves such intent conflicts only before they are translated into service nodes, this document covers intent conflicts and inconsistency issues in a broader sense.

6.2.2. Controller Shield

The Controller Shield project proposes to create a repository called

unified-security plugin (USecPlugin). The unified-security plugin is a general purpose plugin to provide the controller security information to northbound applications. The security information could be for various purposes such as collating source of different attacks reported in southbound plugins and suspected controller intrusions. Information collected at this plugin can also be used to configure firewalls and create IP blacklists for the network.

In terms of security services, the document covers authentication, data integrity, confidentiality, and replay protection. However, the Controller Shield project only covers authentication, data integrity, and replay protection services where the confidentiality service is not considered.

6.2.3. Defense4All

The Defense4All project proposes a SDN application for detecting and mitigating DDoS attacks. The application communicates with ODL controller via the northbound interface and performs the two main tasks; 1) Monitoring behavior of protected traffic and 2) Diverting attacked traffic to selected attack mitigation systems (AMSs).

While the Defense4All project only focuses on defense system at the controller, this document includes broader defense issues at the service node as well as the controller.

6.3. Summary

The verification functions should spreads over the platforms to accomplish the requirements mentioned in clause 3. The correctness of NFV- based services and their network configurations can be checked in the NFV MANO layer which has the entire states of the VNFs. Each NFVI needs to provide verification layer which composed of policy manager, network database and interfaces (e.g. REST APIs). Local properties and invariants can be verified inside the specific NFVI, and the global properties and invariants can be checked by merging local verification results from the related NFVIs.

The verification service provides verification functions to NFV MANO, NFVI, and any other low-level modules such as SDN controllers. For the platform independency, it provides standard APIs to process the verification tasks. It also uses standard APIs provided by OSS such as OpenStack (Neutron) and Open Daylight. The compiler and interpreter translate standard description languages and protocols into the internal model which optimized to the verification tasks. It can process user-defined properties to be checked as well. The properties to be checked whether they are user-defined or pre-defined invariants are managed by property library. The verifier maintains a

set of verification algorithms to check the properties. The network database inside the verification service manages the global network states directly or indirectly.

A PoC can be implemented using OpenStack (Neutron) and Open Daylight. The modules related to verification framework can reside in between network virtualization framework (e.g. OpenStack Neutron) and SDN controller (e.g. Open Daylight). Neutron and Open Daylight uses standard APIs provided by verification service to accomplish verification tasks. The initial use case for the PoC could be, in particular, any of security, performance, etc as mentioned in clause 2.

7. Security Considerations

As already described in clause 2.6, how to verify security holes in VNF FG is very important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, potential security concern should be also carefully checked since several VNFs (e.g., NAT) can modify or update packet headers and payload.

8. Acknowledgements

The authors would like to thank formal methods lab members in Korea University for their verification theory support.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[ETSI-NFV-Arch] ETSI, "Network Function Virtualisation (NFV); Architectural Framework," 2014.

[ETSI-NFV-MANO] ETSI, "Network Function Virtualiztion (NFV) Management and Orchestration," 2014.

[ETSI-NFV-Testing] ETSI, "Network Function Virtualiztion (NFV) Pre-deployment Testing; Report on Validation of NFV

Environments and Services," 2016.

[SIGCOMM-Qazi] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in Proc. ACM SIGCOMM 2013, August 2013.

[ONS-Gember] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: Virtual Middleboxes as First-Class Entities," ONS 2013 and TR.

[SIGCOMM-Gember] A. Gember, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in Proc. ACM SIGCOMM 2014, August 2014.

[HOTSDN-Ghorbani] S. Ghorbani, B. Godfrey, "Towards Correct Network Virtualization", HOTSDN 2014.

[RAFT] https://raftconsensus.github.io/.

[ODL] "OpenDaylight SDN Controller, "http://www.opendaylight.org/

[OPNFV] "Open Platform for NFV, "https://www.opnfv.org/

Authors' Addresses

Myung-Ki Shin
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Ki-Hyuk Nam
Friesty

Email: nam@friesty.com

Sangheon Pack
Korea University

Email: shpack@korea.ac.kr

Seungik Lee
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Ramki Krishnan
Dell

Email: Ramki_Krishnan@dell.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 14, 2017

D. Lopez
TID
J. Bonnet
Altice Labs
M. Peuster
UPB
P. Aranda Gutierrez
UC3M

March 13, 2017

The Role of a Mediation Element in NFV DevOps
draft-sonata-nfvrg-devops-gatekeeper-03

Abstract

This document describes how a mediation element (a "gatekeeper") can be applied to support DevOps practices in the provisioning of network services based on Network Function Virtualisation (NFV).

Table of Contents

1. Introduction

The DevOps model is already an established concept in IT industry reducing time to market by close collaboration between service developers and service operators. The switch to virtualisation technologies in the network and its potential for quicker time-to-market deployment requires the application of agile development cycles supporting a DevOps approach. This kind of approach will overcome key inhibitors that network operators face when deploying NFV, such as lack of legacy compatibility, resource orchestration, automation and multi-vendor interoperability, hence facilitating the transition to a software-driven network. The adoption of the DevOps model for network services will contribute to interaction between development, testing, and operation of network functionalities and network services. Both the function/service description formats as well as the infrastructure resource descriptions will be able to express and use legacy cases, e.g., the case of a non-virtual network function bound to a specific place in the network, with the data flows routed accordingly.

Network Service Providers (NSPs) must be able to orchestrate diverse network functions from multiple sources for automation and streamline them into an inter-organizational DevOps workflow. To embrace the DevOps model implies not only to shorten time between deploying, testing and validating of services, but also to enable the mechanisms for the network to consider application layer requirements and reaction to SLAs, and to ease network reconfiguration in order to achieve fast reaction in a timely manner.

Development and operational tools, the two essential pillars of DevOps, translate into the need of addressing the interfacing of service development tasks and the service platform, which in DevOps are closely linked together. It is required to emphasize the need for quick turn-around times in service development and operation, and materialize it in a mediated interface making a direct collaboration on both the development and the platform side possible.

The branching to multiple stakeholders in the service lifecycle creates an inter-organizational dynamics that must be taken into account. A realistic NFV DevOps approach has to take into account a trustworthy cycle with a mediation element that ensures compliance policies set by the NSP considering legacy situation, allowing developers across stakeholders to enter the ecosystem. Such a mediation element is what we will refer as a "gatekeeper" in the rest of this document. The resulting strategy opens collaborating opportunities while mitigating liability risks across the network service lifecycle.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. The Essential Components for NFV DevOps

The collaboration between the development and operational tasks to build a service lifecycle according to the DevOps principles requires to combine service programming and orchestration frameworks by means of the following components:

- Catalogues, storing static information regarding network functions and services: code, executables, configuration data, and specific management requirements and preferences. Contents, location, organization, and implementation of catalogues for different artefacts can vary considerably. However, users of these catalogues need to deal with them in a consistent fashion and the differences across different catalogues need to be harmonized and abstracted away. As a high-level categorization, the following three types of catalogues can be considered:
  - Private catalogues of service developers, where they can define, access, reuse, and modify services and service components.
  - Service platform catalogues made available to authorized service developers for reusing existing components in their services, and used for storing services and their components that need to be deployed by the service platform.
  - Public catalogues storing artefacts developed and maintained by third-party developers on arbitrary platforms accessible to service developers and service platform operators.
- Service Development Kit (SDK). The SDK supports service developers by providing a service programming model and a development tool-chain, designed to support developers in defining and testing complex services consisting of multiple network functions, and to facilitate custom implementations of individual network functions. The tools of this tool-chains provides all necesaray interfaces to establish fully automated continuous

integration (CI) workflows. The implemented artefacts are stored in the developer's private catalogues. Moreover, service components can easily be obtained from external catalogues. The obtained artefacts can be directly used in a service or after being modified and tested using the SDK development tools. The service components and all the information necessary for deployment and execution of a service are bundled together into a package. The service package can be handed over to a service platform for actual deployment and for testing, debugging, and profiling purposes. The tools provided by a service development tool-chain can be classified as follows:

* Pre-validation tools used by service developers to ensure the correctness of service and function descriptions, including syntax checks, static structure validation, and integrity ensurance tools.

* Offline testing and emulation tools used by service developers to conduct functional tests of complex services in well-defined, reproducible environments. Especially focusing on the integration and interoperability between multiple network functions combined to a single service.

* Packaging tools responsible to create the final service bundle. This class includes tools that automatically resolve external dependencies of a service to automatically include required artifacts into a package. It also contains tools which sign the final package to give service platforms the necessary confidence that service packages have not been altered during the uploading procedure and to ensure the authenticity of the package.

* Tools to support the interaction of a developer with service platforms as well as services and functions deployed in these platforms. This includes two ways of interactions. First, uploading, instantiation and management of service packages on service platforms. Second, receiving runtime, debugging, and monitoring information from the platform as well as accessing artifacts stored in platform catalogues.

o Service Platform. The service platform receives the service packages implemented and created with the help of the SDK and is responsible for placing, deploying, provisioning, scaling, and managing the services on existing cloud infrastructures. It can also provide direct feedback about the deployed services to the SDK, for example, monitoring data about a service or its components. The service developer can ship the service package to the service platform together with service- or function-specific

lifecycle management requirements and preferences. A gatekeeper module in the service platform is responsible for processing the incoming and outgoing requests.

- Underlying Infrastructure. The infrastructure needs to host and execute the actual network functions of a service, e.g., as a virtual machine. The service platform sends necessary information and instructions for execution and lifecycle management of services to the infrastructure. The infrastructure may belong to the service platform operator, or to a third-party infrastructure operator. The interaction between the service platform and the infrastructure is done through a Virtual Infrastructure Manager (VIM). In a typical deployment, the service platform runs directly on top of an actual infrastructure. However, there can be service platforms supporting a recursive deployment model, where a service platform can act as an abstraction to the underlying infrastructure for another service platform. The service platform gatekeeper can play a relevant role to support mediated recursion as well.

The DevOps workflow is supported by the integration between the SDK and the service platform. This workflow implies continuous deployment and continuous integration during service development. The main entity exchanged between the SDK and the service platform is the service package to be deployed and runtime information like monitoring data and performance measurements regarding the service package, which is provided to the service developer during the development phase, as well as the runtime. This information can be used for optimizing, modifying, and debugging the operation and functionality of services.

4. The Role of the Gatekeeper

The gatekeeper is the module in the service platform that mediates the interactions between the SDK and the SP, settling the development and operational tasks, by performing the basic functions described here.

4.1. User Management

User management allows the service platform owner to control who can do what in the platform. This feature is particularly important in recursive scenarios, on which we may have a chain of service platforms interacting for the implementation of an end-to-end service.

The most basic feature of any user management component will be to

know who is the user, a feature that is usually called authentication. Authentication requires user registration and the maintenance of user identity attributes, including not only identification attributes (user identifiers, passwords, public keys, trusted signing certificates, etc.) but also other information supporting different authorization schemas, such as group-based or role-based ones.

The definition of what each (known) user can do is usually called authorization. The most common approach nowadays to authorization is called role-based, in which each user is assigned one (or more) role(s) and different roles have different permissions. This extra level of indirection, that is users to roles and roles to permissions, simplifies the overall maintenance of the system, when compared to a more direct scheme, like users permissions. Specially when accessing external APIs, it is common to issue temporary keys (then usually called tokens) which enable temporary access to those APIs. Real keys therefore do not leave the realm on which they are valid and useful, thus increasing the overall level of security.

To support a DevOps environment the following roles are considered:

- Developer, able to publish and update service packages on the service platform through the SDK, as well as other operations related to service package status.

- Service provider, in charge of structuring and managing the services available for a certain organization, or organizational group, defining an administrative domain.

- Customer, as a user of the public services available for the administrative domain they belong to, managing their lifecycles (instantiating, pausing, resuming, retiring...).

- Service platform admin, with management capabilities on the platform itself, as well as superuser-like control over the available services. These capabilities include the registration of roles and users, and the association of users to roles, enabling the authentication and authorization mechanisms described above.

4.2. Package Management

The gatekeeper receives the software to be validated in the form of packages. Package management is mostly about accepting and validating new or updated packages. The metadata describing such packages is called package descriptor, and constitutes the core of the gatekeeper interface.

Only known (i.e., successfully authenticated) and authorized users will be able to submit new or revised services through the gatekeeper. On-boarding of a package can only be considered successful when package validation and attestation is successful. Only then the (new version of) the package will become part of the catalogue. On-boarding requests are usually processed in a first come, first served way, otherwise contradictory requests may jeopardize the whole system. The usual solution for this problem is to use a queue mechanism that guarantees this sequence.

A package descriptor is validated in several ways:

- Syntax, comprising the validation against the expected package descriptor format.

- Semantics, which includes the validation of at least the basic parameters. The exact semantic aspects to be validated will depend on the content and format chosen for the package descriptor.

- Licensing, by checking that all external dependencies (i.e., packages, libraries or services) have to have their licenses checked before being used.

- Tests availability. Although this might be seen as part of the syntactic/semantic correction, there must be a set of tests that can be executed when validating the package. Depending of the scope and complexity of the service, these tests may be a subset of the unit tests or a more elaborate suit of integration tests.

- Tests execution. Besides providing a suit of tests, these have to be successfully executed. This execution may (usually will) imply the creation and initialization of at least one test environment. When the package under test depends on other packages, libraries or services, those too should be taken into account in the execution of the package tests.

The service package must include signatures, generated by the SDK’s packaging tools, that allow the validation of the integrity and authenticity of the package’s contents, the component VNFs, and other components (forwarding graphs, test suites, etc.). These signatured can be optionally used to attest the components at different stages of their lifecycle, and/or during runtime.

Requests for a change in the life-cycle of a package must be validated. This might be a simple authorization configuration.

- Deployment. Valid packages, available at the service platform repository, may receive a request for deployment. Package deployment implies the creation of all the environments and connections needed for the package and its dependencies to work and of an instance of that package.

- Instance (re)-configuration. A deployed package instance may need to be configured. A special kind of configuration might be, for packages supporting multi-tenancy, adding a new tenant. The package may have "open parameters" that can only be closed upon instantiation (e.g., an IP address). If a Package upgrade happens, a reconfiguration of the instance must also be made.

- Instance (re-)start. When, e.g., configuration changes.

- Instance monitoring. This is not strictly a change in the life-cycle, but would require the execution of certain aspects identified by the package descriptor or its components.

- Instance stop. Includes soft-stop (i.e., not accepting new requests and letting currently running request reach their end of life normally, with a pre-defined time-out) and hard-stop (i.e., a sudden stop, with requests still being answered by the service).

- Instance termination. Frees any resource(s) that were being used, taking care of dependencies.

- Removal. It requieres an evaluation of currently running instances and dependencies.

4.3. Monitor Data Transfer

The gatekeeper is the first point of access to reach the SP from the SDK. Service developers can use their identities from the SDK to access monitor data from the SP. After the successful AuthN/AuthZ phase, developers are granted a session token to access monitoring data. Multiple developers will use different data access views to get their own set of authorized monitor data.

It is desirable that the gatekeeper is transparent to the monitor data transfer, acting as a pure forwarder, apart from the AuthN/AuthZ phase. Optionally, the gatekeeper could filter non-numerical monitored data (e.g. obfuscate domain names, IP/MAC addresses...) transferred in logfiles or packet streams. The session token is used by the monitor data management components to decide on which data to expose, so metrics of another user, other services not started by the developer or the SP itself can never be queried by the SDK. In addition, other limits can be enforced, such as:

- Limit the number of monitor samples
- Limit the data size to be received
- Limit the time frame during which metrics are accessible

5. Security Considerations

The gatekeeper acts as the security enforcement point for all DevOps interactions between the development and operational tasks, and even between different layers in recursive structure.

Gatekeeper APIs will have to be secured, providing identification, confidentiality, integrity and non-repudiation.

Other potential threats are related to denial-of-service, whereby an adversary could make the whole NFV environment unusable by overloading the gatekeeper with a high number of requests or requests tailored to exhaust its resources. Mechanisms for overload detection and mitigation should be put in place.

6. IANA Considerations

This document requires no IANA actions.

7. Acknowledgements

This work has been partially performed in the scope of the SONATA project [SONATA], which has received funding from the European Union's Horizon 2020 research and innovation programme. The authors would like to acknowledge the contributions of their colleagues. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[SONATA] "Project SONATA", <http://www.sonata-nfv.eu/>.

Authors' Addresses

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Jose Bonnet
Altice Labs
Rua Eng. Jose Ferreira Pinto Basto
Aveiro, 3810-106
Portugal

Phone: +351 234 403 200
Email: jbonnet@alticelabs.com

Manuel Peuster
Paderborn University
Warburgerstrasse 100
Paderborn, 33098
Germany

Phone: +49 5251 60 4341
Email: manuel.peuster@upb.de

Pedro A. Aranda Gutierrez
UC3M

Email: paaguti@hotmail.com