NFVRG
Internet-Draft
Intended status: Informational
Expires: August 31, 2018

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

February 27, 2018

High-level VNF Descriptors using NEMO
draft-aranda-nfvrg-recursive-vnf-05

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. An implementation generating VNF Descriptors (VNFDs) for OpenMANO and OSM is available.

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 https://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 August 31, 2018.

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

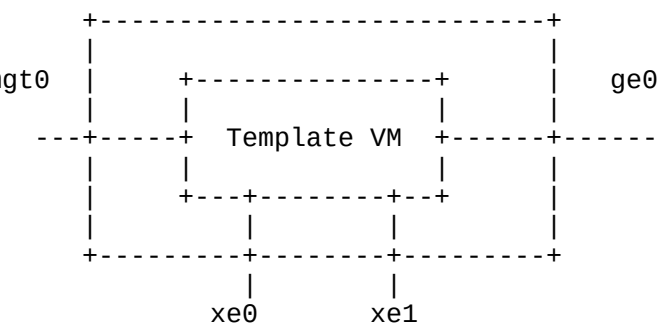
Virtual network function descriptors (VNFDs) are used in the Management and orchestration (MANO) framework of the ETSI NFV to achieve the optimal deployment of virtual network functions (VNFs). The Virtual Infrastructure Manager (VIM) uses this information to place the functions optimally. VNFDs include 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.

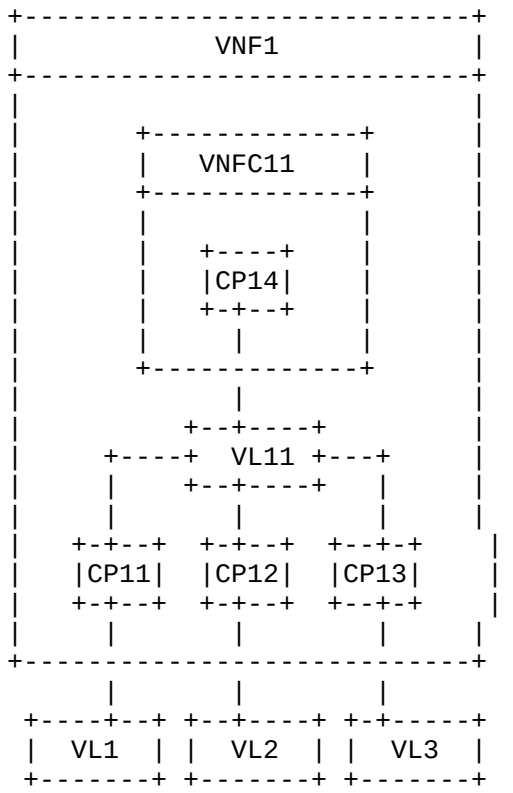


Figure 2: VNF example

```
########################################
# 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:

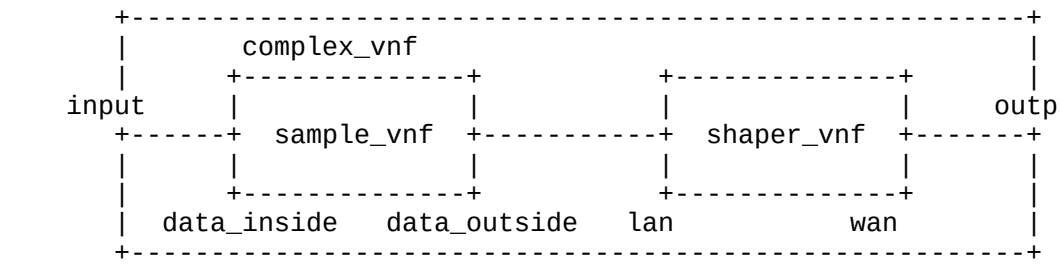


Figure 6

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. Implementation

There is a proof of concept implementation of the concepts described in this draft is available at github [5]. This proof of concept is implemented as an OpenDayLight (ODL) [6] plugin and includes two output stages to generate VNFDs for OpenMANO and OSM. In its current implementation, the ODL plugin depends on an outdated NEMO project.

6. Future work

Future work includes an implementation that does not depend on ODL and extensions to the language to separate control and data plane connections explicitly.

7. 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 [7]) can also be used as the lowest level in this extension to the NEMO language.

8. IANA Considerations

This draft includes no request to IANA.

9. 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.

10. 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).

11. References

11.1. 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, <https://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, <https://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.

11.2. 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.

11.3. 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] https://github.com/telefonicaid/vibnemo

[6] http://www.opendaylight.org

[7] 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

Email: elena.batanero.18@gmail.com

NFV RG
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

CJ. Bernardos, Ed.
UC3M
LM. Contreras
TID
I. Vaishnavi
Huawei
R. Szabo
Ericsson
J. Mangues
CTTC
X. Li
NEC
F. Paolucci
A. Sgambelluri
B. Martini
L. Valcarenghi
SSSA
G. Landi
Nextworks
D. Andrushko
MIRANTIS
A. Mourad
InterDigital
March 5, 2018

Multi-domain Network Virtualization
draft-bernardos-nfvrg-multidomain-04

Abstract

This document 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 and 5G-TRANSFORMER projects.

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 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.

Furthermore, 5G networks are being designed to be capable of fulfilling the needs of a plethora of vertical industries (e.g., automotive, eHealth, media), which have a wide variety of requirements [ngmn_5g_whitepaper]. The slicing concept tries to make the network of the provider aware of the business needs of tenants (e.g., vertical industries) by customizing the share of the network assigned to them. The term network slice was coined to refer to a

complete logical network composed of network functions and the resources to run them [ngmn_slicing]. These resources include network, storage, and computing. The way in which services requested by customers of the provider are assigned to slices depends on customer needs and provider policies. The system must be flexible to accommodate a variety of options.

Another characteristic of current and future telecommunication networks is complexity. It comes from three main aspects. First, heterogeneous technologies are often separated in multiple domains under the supervision of different network managers, which exchange provisioning orders that are manually handled. This does not only happen between different operators, but also inside the network of the same operator. Second, the different regional scope of each operator requires peering with others to extend their reach. And third, the increasing variety of interaction among specialized providers (e.g., mobile operator, cloud service provider, transport network provider) that complement each other to satisfy the service requests from customers. In conclusion, realizing the slicing vision to adapt the network to needs of verticals will require handling multi-provider and multi-domain aspects.

Additionally, Network Function Virtualization (NFV) and Software Defined Networking (SDN) are changing the way the telecommunications sector will deploy, extend and operate its networks. Together, they bring the required programmability and flexibility. Moreover, these concepts and network slicing are tightly related. In fact, slices may be implemented as NFV network services. However, building a complete end-to-end logical network will likely require stitching services offered by multiple domains from multiple providers. This is why multi-domain network virtualization is crucial in 5G networks.

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.

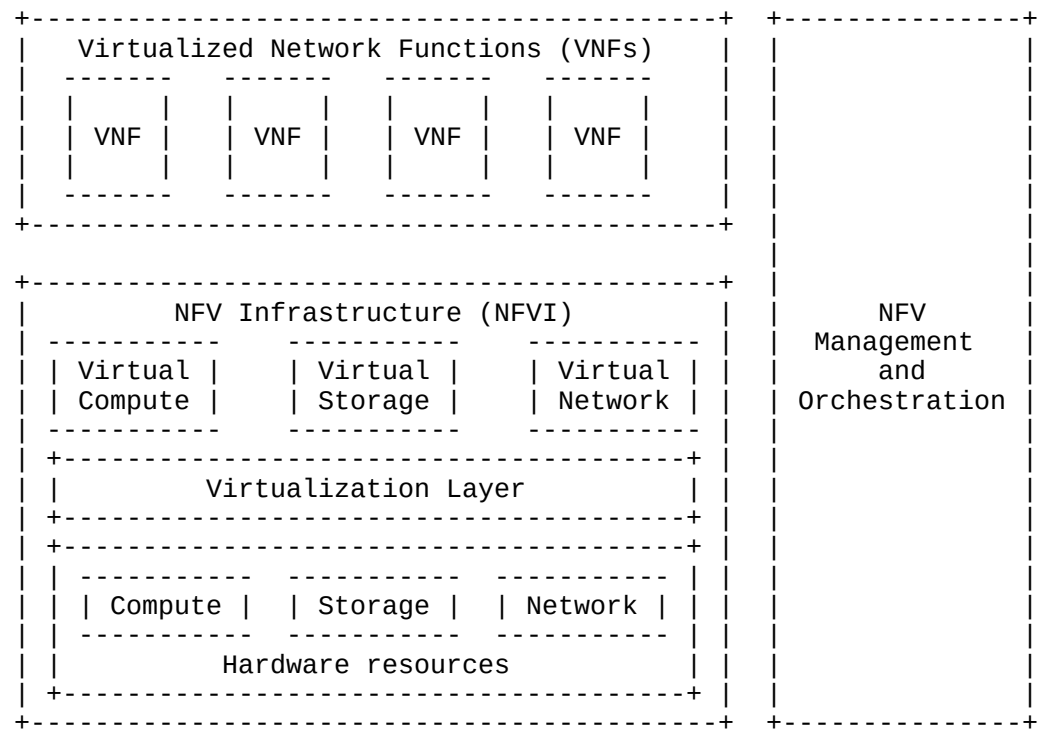


Figure 1: ETSI NFV framework

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

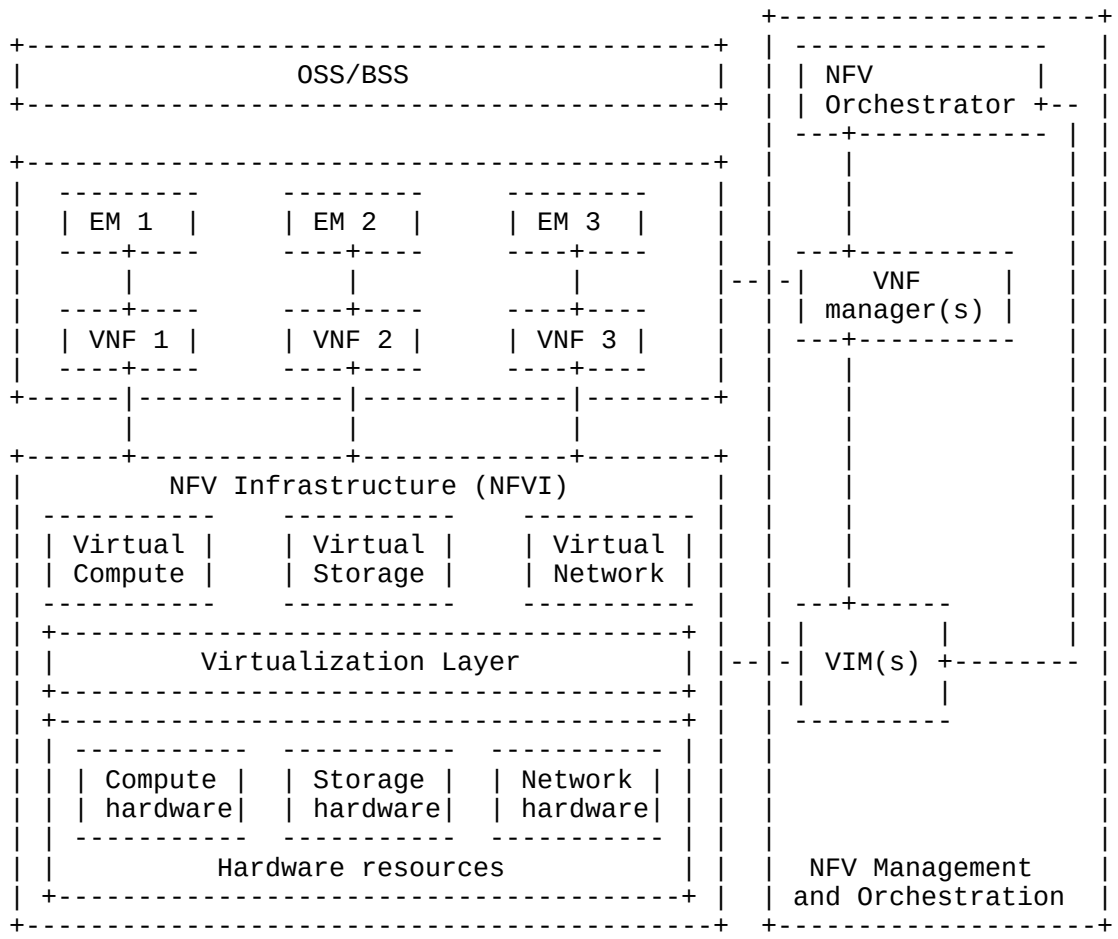


Figure 2: ETSI NFV reference architecture

4. Multi-domain 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.

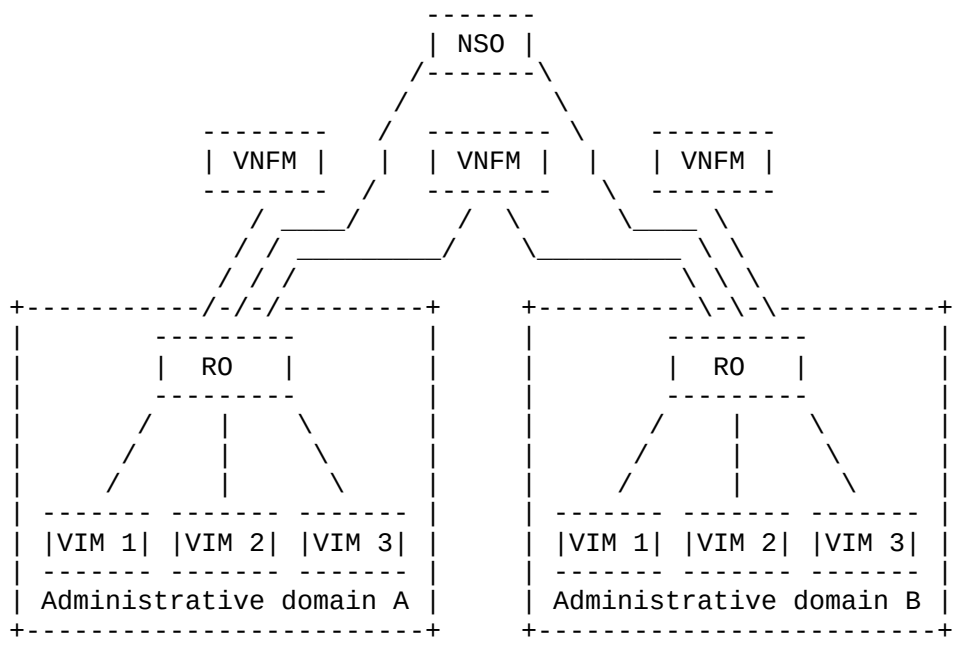


Figure 3: Infrastructure provided using multiple administrative domains (from ETSI GS NFV-IFA 009 V1.1.1)

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.

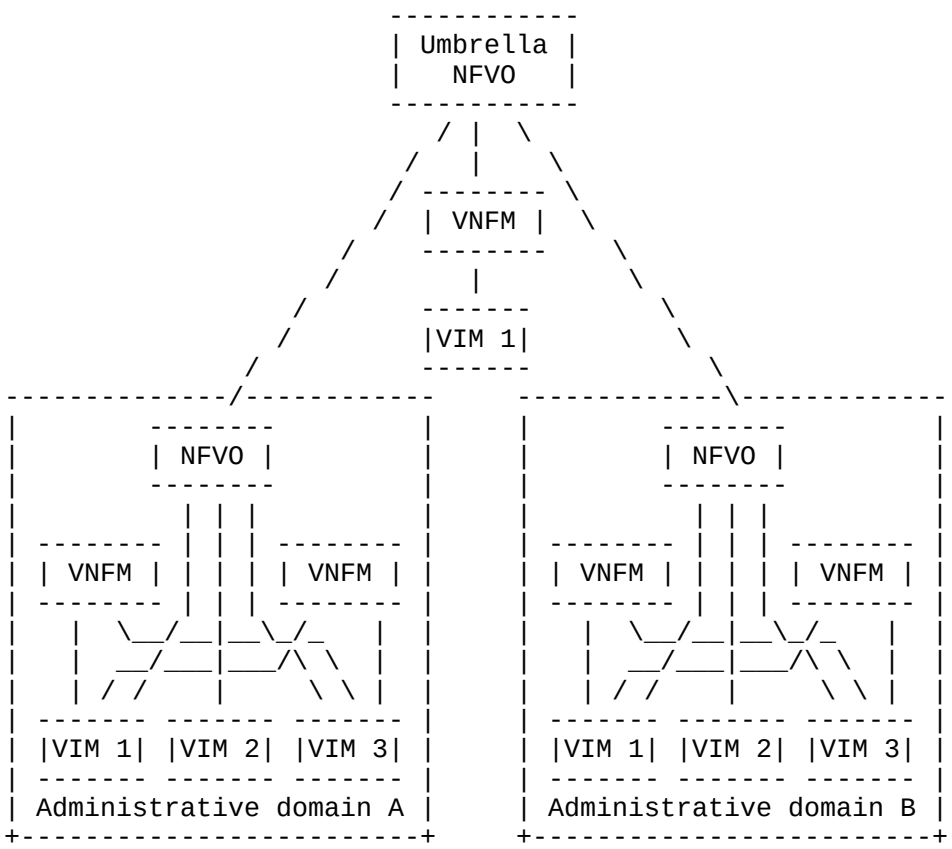


Figure 4: Network services provided using multiple administrative domains (from ETSI GS NFV-IFA 009 V1.1.1)

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.

In January 2018 ETSI NFV released a report about NFV MANO architectural options to support multiple administrative domains [etsi_nvf_ifa028]. This report presents two use cases: the NFVI as a Service (NFVIaaS) case, where a service provider runs VNFs inside an NFVI operated by a different service provider, and the case of Network Services (NS) offered by multiple administrative domains, where an organization uses NS(s) offered by another organization.

In the NFVIaaS use case, the NFVIaaS consumer runs VNF instances inside an NFVI provided by a different service provider, called NFVIaaS provider, that offers computing, storage, and networking resources to the NFVIaaS consumer. Therefore, the NFVIaaS consumer has the control on the applications that run on the virtual resources, but has not the control of the underlying infrastructure, which is instead managed by the NFVIaaS provider. In this scenario, the NFVIaaS provider’s domain is composed of one or more NFVI-PoPs and VIMs, while the NFVIaaS consumer’s domain includes one or more NSs and VNFs managed by its own NFVO and VNFMs, as depicted in Figure 5.

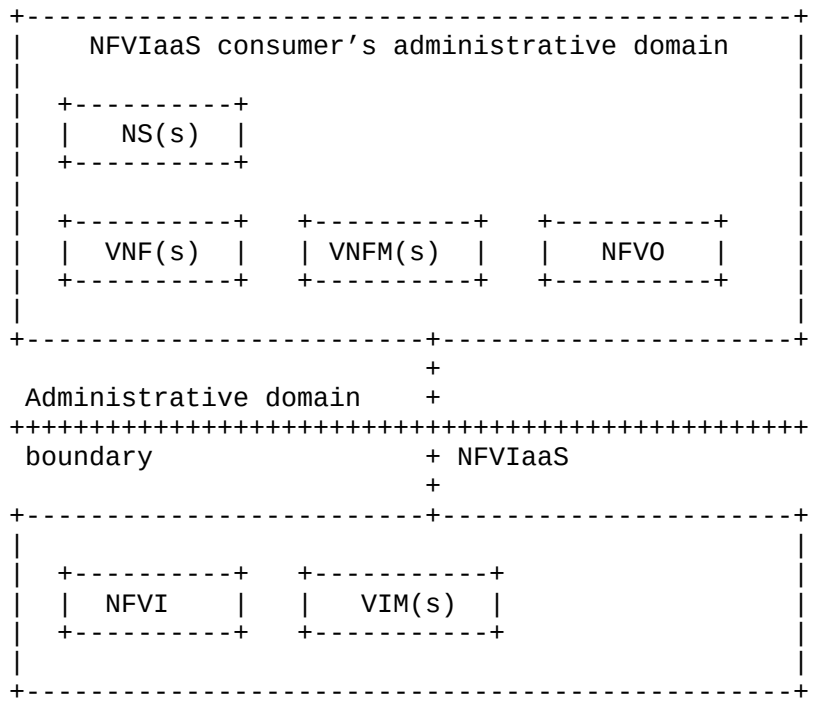


Figure 5: NFVI use case

The ETSI IFA 028 defines two main options to model the interfaces between NFVIaaS provider and consumer for NFVIaaS service requests, as follows:

1. Access to Multiple Logical Points of Contacts (MLPOC) in the NFVIaaS provider's administrative domain. In this case the NFVIaaS consumer has visibility of the NFVIaaS provider's VIMs and it interacts with each of them to issue NFVIaaS service requests, through Or-Vi (IFA 005) or Vi-Vnfm (IFA 006) reference points.

2. Access to a Single Logical Point of Contact (SLPOC) in the NFVIaaS provider's administrative domain. In this case the NFVIaaS provider's VIMs are hidden from the NFVIaaS consumer and a single unified interface is exposed by the SLPOC to the NFVIaaS consumer. The SLPOC manages the information about the organization, the availability and the utilization of the infrastructure resources, forwarding the requests from the NFVIaaS consumer to the VIMs. The interaction between SLPOC and NFVIaaS consumer is based on IFA 005 or IFA 006 interfaces, while

the interface between the SLPOC and the underlying VIMs is based on the IFA 005.

The two options are shown in Figure 6 and Figure 7 respectively, where we assume the direct mode for the management of VNF resources. In addition, the ETSI IFA 028 includes the possibility of an indirect management mode of the VNF resources through the consumer NFVIaaS NFVO and the IFA 007 interface. In this latter case between the consumer NFVIaaS NFVO and the provider NFVIaaS NFVO only the IFA 005 interface is utilized.

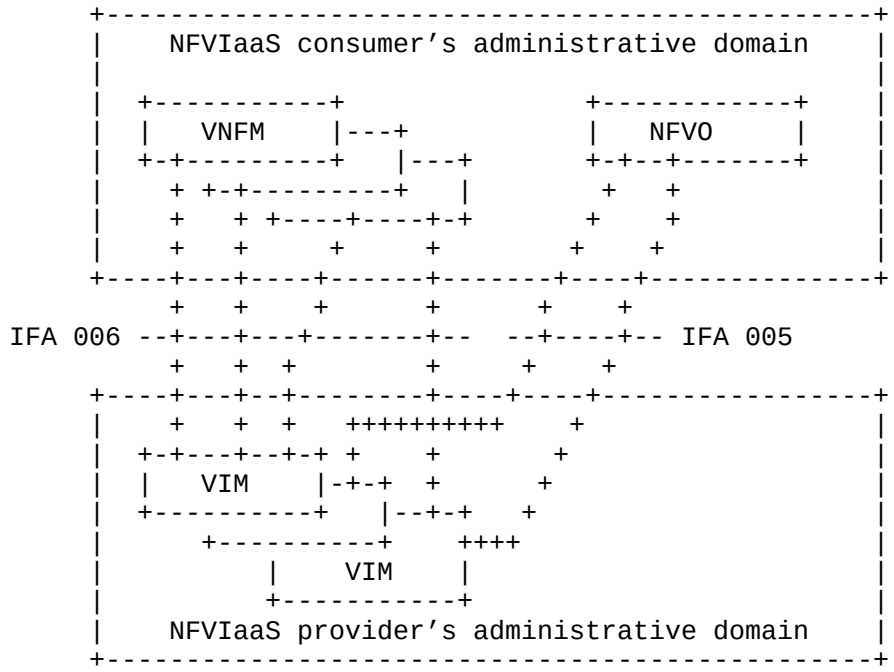


Figure 6: NFVIaaS architecture: MLPOC option

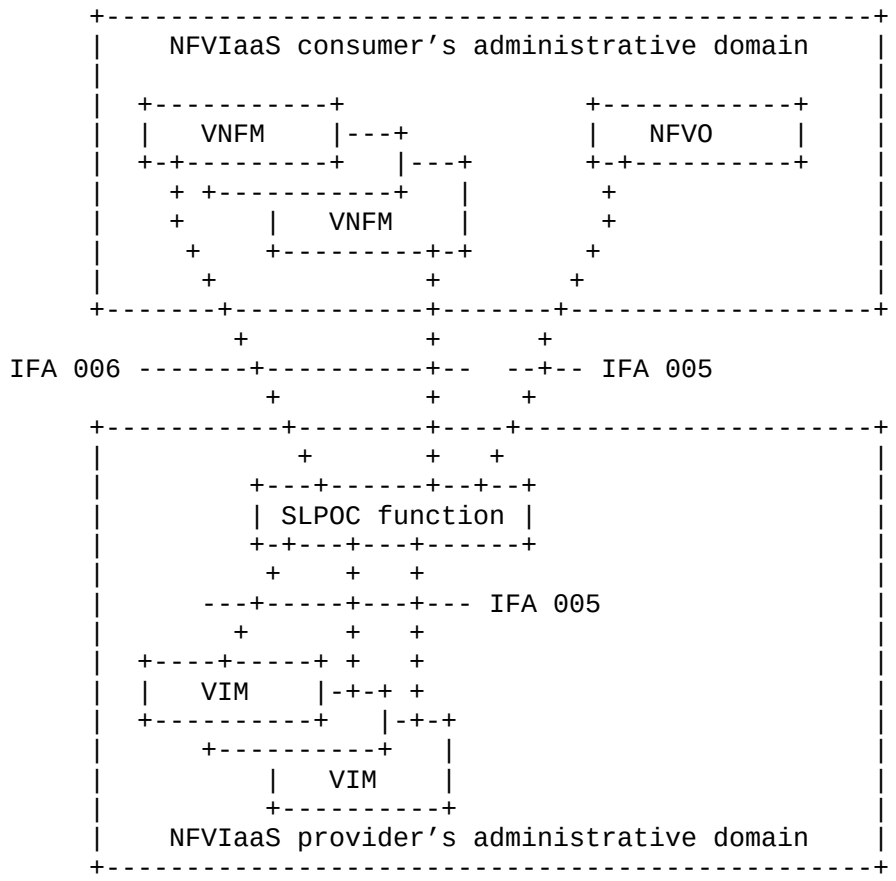


Figure 7: NFVIaaS architecture: SLPOC option

In the use case related to Network Services provided using multiple administrative domains, each domain includes an NFVO and one or more NFVI PoPs, VIMs and VNFMs. The NFVO in each domain offers a catalogue of Network Services that can be used to deploy nested NSs, which in turn can be composed into composite NSs, as shown in Figure 8. Nested NSs can be also shared among different composite NSs.

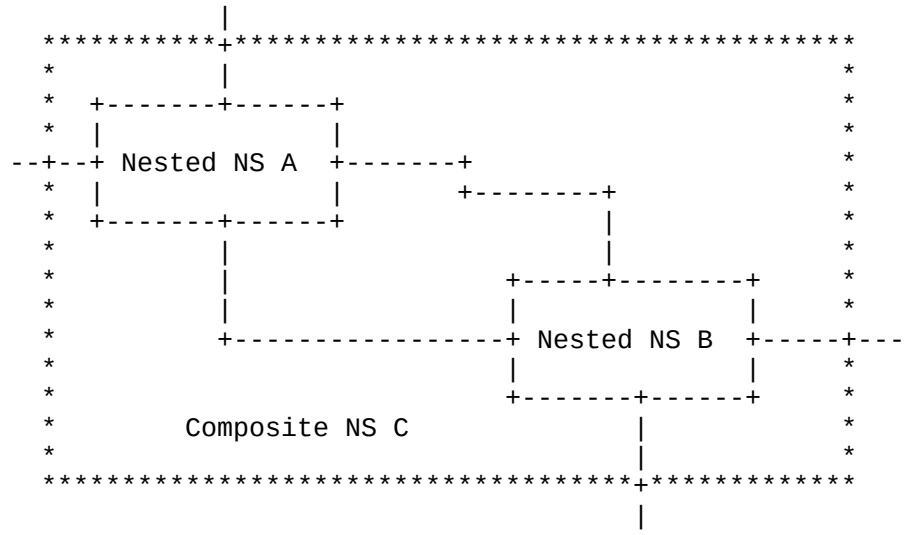


Figure 8: Composite and nested NSs

The management of the NS hierarchy is handled through a hierarchy of NFVOs, with one of them responsible for the instantiation and lifecycle management of the composite NS, coordinating the actions of the other NFVOs that manage the nested NSs. These two different kinds of NFVOs interact through a new reference point, named Or-Or, as shown in Figure 9, where NFVO-1 manages composite NSs and NFVO-2 manages nested NSs. To build the composite NSs, the responsible NFVO consult its own catalogue and may subscribe to the NSD notifications sent by other NFVOs.

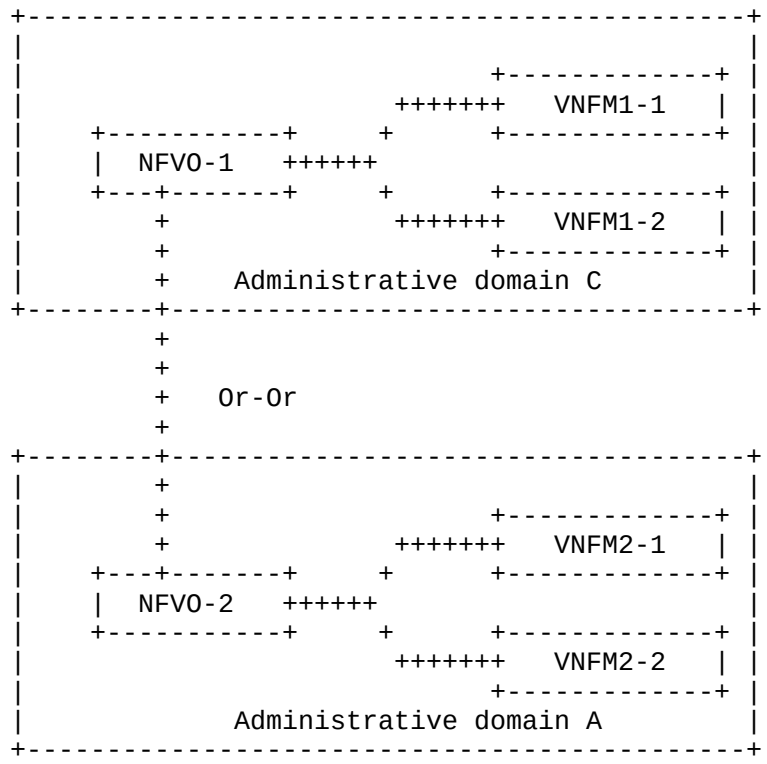


Figure 9: Architecture for management of composite and nested NS

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 10 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.

Some questions arise from this. It would be good 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?

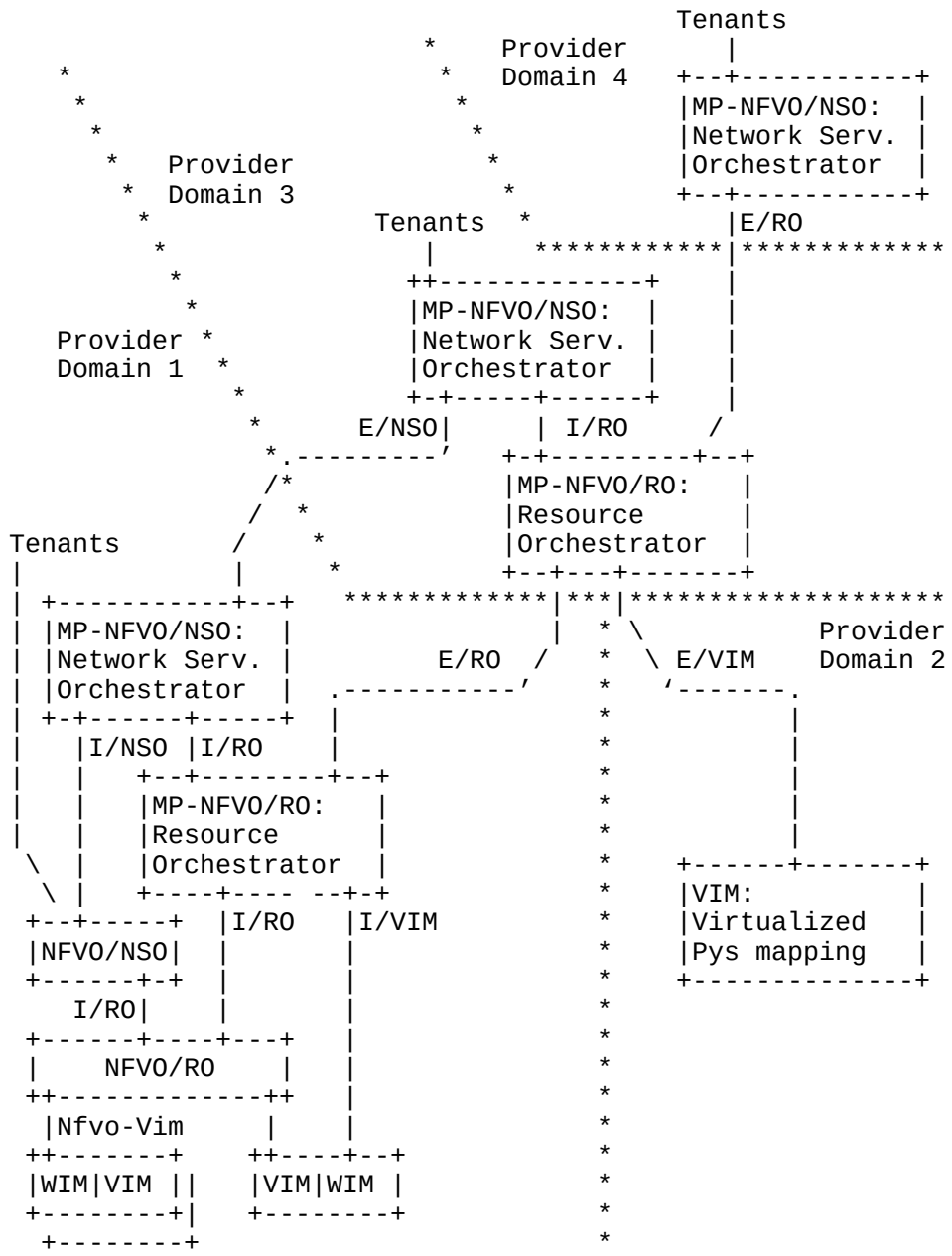


Figure 10: NSO-RO Split: possible multi-provider APIs - an illustration

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 11.

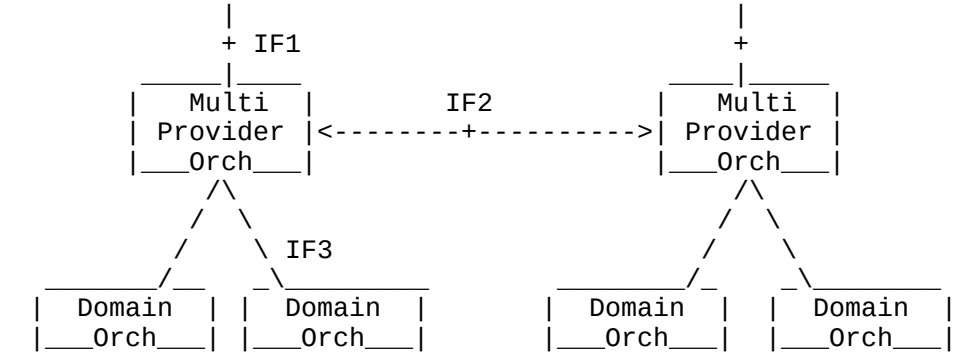


Figure 11: Multi Domain Multi Provider reference architecture

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 tenant 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 recommending to the orchestration engine the reconfiguration of the service or slice which again needs to be performed over IF2 and IF3.

Each of the three steps is assigned to a specific block in our high level architecture shown in Figure 12.

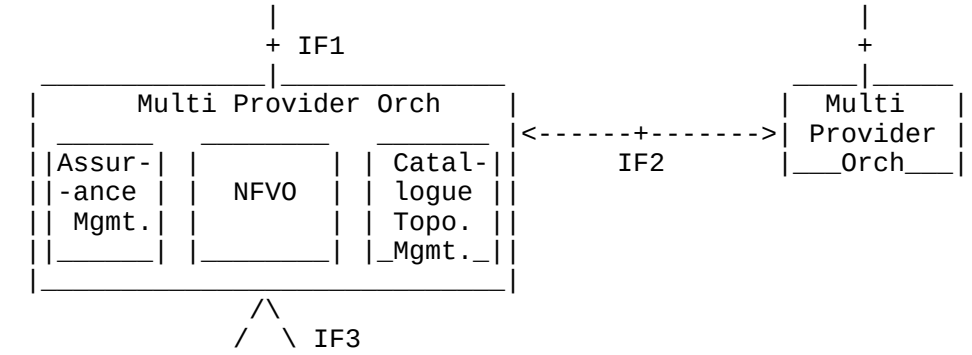


Figure 12: Detailed MPO reference architecture

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 by the 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 management 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 recommends reconfigurations to the NFVO.

6.1. Interworking interfaces

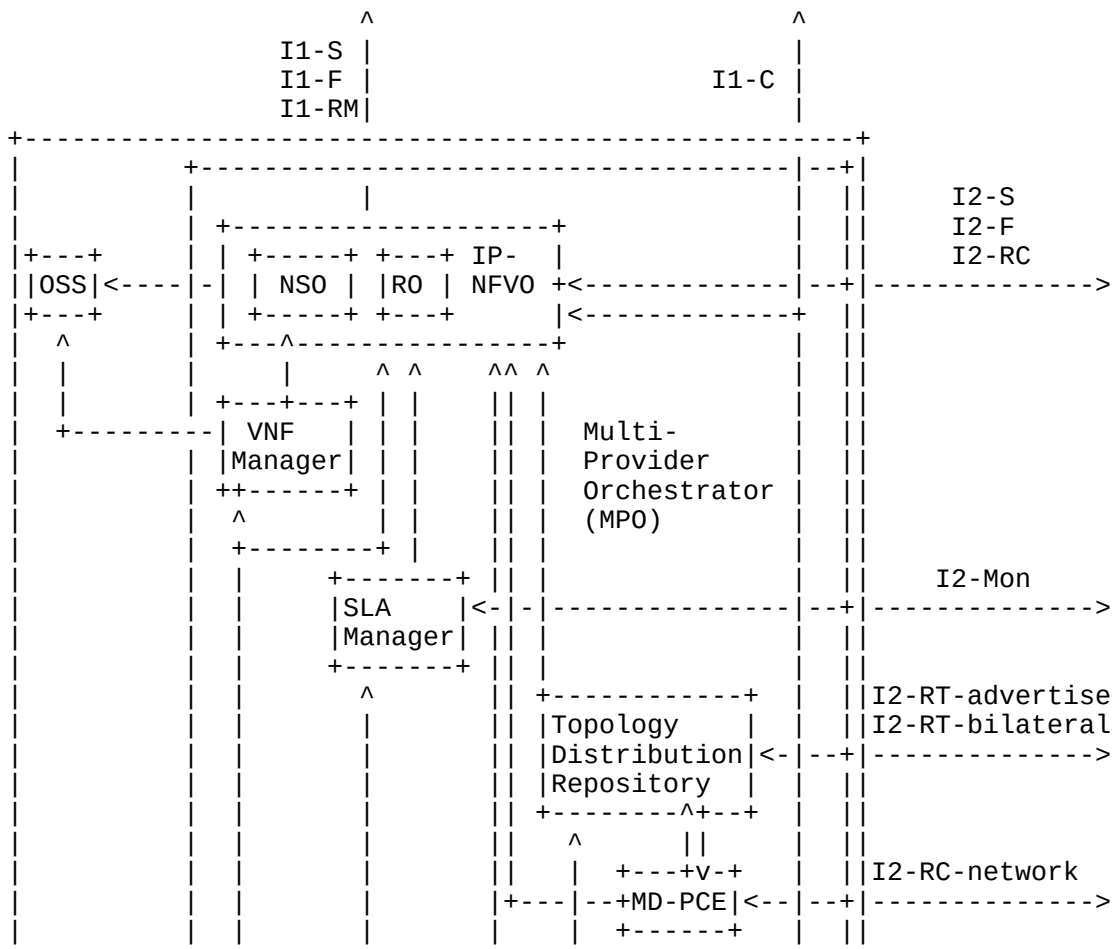
In this section we provide more details on the interworking interfaces of the MPO reference architecture. Each interface IF1, IF2 and IF3 is broken down into several sub-interfaces. Each of them has a clear scope and functionality.

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.

6.2. 5GEx Multi Architecture

The 5G-PPP H2020 5GEx projects addresses the proposal and the deployment of a complete Multi-Provider Orchestrator providing, besides network and service orchestration, service exposition to other providers. The main assumptions of the 5GEx functional architecture are a) a multi-operator wholesale relationship, b) a full multi-vendor inter-operability and c) technology-agnostic approach for physical resources. The proposed functional architecture of the 5GEx MPO is depicted in Figure 13.



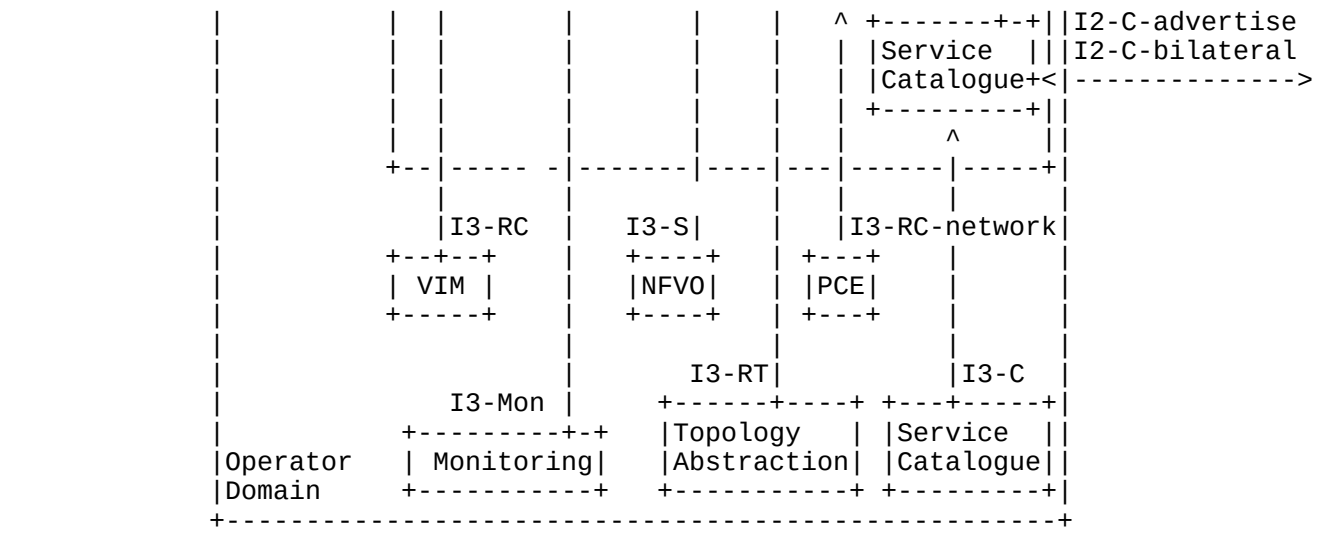


Figure 13: 5GEx MPO functional architecture

Providers expose MPOs service specification API allowing OSS/BSS or external business customers to perform and select their requirements for a service. Interface I1-x is exploited as a northbound API for business client requests. Peer MPO-MPO communications implementing multi-operator orchestration operate with specific interfaces referred to as I2-x interfaces. A number of I2-based interfaces are provided for communication between specific MPO modules: I2-S for service orchestration, I2-RC for network resource control, I2-F for management lifecycle, I2-Mon for inter-operator monitoring messages, I2-RT for resource advertisement, I2-C for service catalogue exchange, I2-RC-network for the QoS connectivity resource control. Some I2 interfaces are bilateral, involving direct relationship between two operators, and utilized to exchange business/SLA agreements before entering the federation of inter-operator orchestrators. Each MPO communicates through a set of southbound interface, I3-x, with local orchestrators/controllers/VIM, in order to set/modify/release resources identified by the MPO or during inter-MPO orchestration phase. A number of I3 interfaces are defined: I3-S for service orchestration towards local NFVO, I3-RC for resource orchestration towards local VIM, I3-C towards local service catalogue, I3-RT towards local abstraction topology module, I3-RC-network towards local PCE or network controller, I3-Mon towards local Resource Monitoring agent. All the considered interfaces are provided to cover either flat orchestration or layered/hierarchical orchestration. The possibility of hierarchical inter-provider MPO interaction is enabled at a functional level, e.g., in the case of

operators managing a high number of large administrative domains. The main MPO modules are the following:

- The Inter-provider NFVO, including the RO and the NSO, implementing the multi-provider service decomposition
- the VNF/Element manager, managing VNF lifecycle, scaling and responsible for FCAPS (Fault, Configuration, Accounting, Performance and Security management)
- the SLA Manager, in charge of reporting monitoring and performance alerts on the service graph
- the Service Catalogue, exposing available services to external client and operators
- the Topology and Resource Distribution module and Repository, exchanging operators topologies (both IT and network resources) and providing abstracted view of the own operator topology
- the Multi-domain Path Computation Element (PCE implementing inter-operator path computation to allow QoS-based connectivity serving VNF-VNF link).

The Inter-provider NVFO selects providers to be involved in the service chained request, according to policy-based decisions and resorting to Inter-Provider topologies and service catalogues advertised through interfaces I2-RT-advertise and I2-C-advertise, respectively. Network/service requests are sent to other providers using the I2-RC and I2-S interfaces, respectively. Policy enforcement for authorized providers running resource orchestration and lifecycle management are exploited through interfaces I2-RC and I2-F, respectively. The VNF/Element Manager is in charge of managing the lifecycle of the VNFs part of the services. More specifically, it is in charge to perform: the configuration of the VNFs, also in terms of security aspects, the fault recovery and the scaling according to their performance. The SLA Manager collects and aggregates quality measurement reports from probes deployed by the Inter-Provider NFVO as part of the service setup. Measurements results at the Manager represent aggregated results and are computed and stored utilizing the I2-Mon interface between Inter-Provider MPOs sharing the same service. Faults and alarms are moreover correlated to raise SLA violation to remote inter-provider MPOs and, optionally, to detect the source and the location of the violation, triggering service re-computation/rerouting procedures. The Service Catalogue stores information on network services and available VNFs and uses I2-C interfaces (either bilateral or advertised) to advertise and updating such offered services to other operators. To enable inter-

provider service decomposition, multi-operator topology and peering relationships need to be advertised. Providers advertise basic inter-provider topologies using the I2-RT-advertse interface including, optionally, abstracted network resources, overall IT resource capabilities, MPO entry-point and MD-PCE IP address. Basic advertisement takes place between adjacent operators. These information are collected, filtered by policy rules and propagated hop-by-hop. In 5GEx, the I2-RT-advertise interfaces utilizes BGP-LS protocol. Moreover, providers establish point-to-point bilateral (i.e., direct and exclusive) communications to exchange additional topology and business information, using the I2-RT-bilateral interface. Service decomposition may imply the instantiation of traffic-engineered multi-provider connectivity, subject to constraints such as guaranteed bandwidth, latency or minimum TE metric. The multi-domain PCE (MD-PCE) receives the connectivity request from the inter-provider NFVO and performs inter-operator path computation to instantiate QoS-based connectivity between two VNFs (e.g., Label Switched Paths). Two procedures are run sequentially:

operators/domain sequence computation, based on the topology database, provided by Topology Distribution module, and on specific policies (e.g., business, bilateral),

per-operator connectivity computation and instantiation.

In 5GEx, MD-PCE is stateful (i.e., current connectivity information is stored inside the PCE) and inter-operator detailed computation is performed resorting to the stateful Backward Recursive PCE-based computation (BRPC) [draft-stateful-BRPC], deploying a chain of PCEP sessions among adjacent operators, each one responsible of computing and deploying its segment. Backward recursive procedure allows optimal e2e constrained path computation results.

6.3. 5G-TRANSFORMER Architecture

5G-TRANSFORMER project proposes a flexible and adaptable SDN/NFV-based design of the next generation Mobile Transport Networks, capable of simultaneously supporting the needs of various vertical industries with diverse range of requirements by offering customized slices. In this design, multi-domain orchestration and federation are considered as the key concepts to enable end-to-end orchestration of services and resources across multiple administrative domains.

The 5G-TRANSFORMER solution consists of three novel building blocks, namely:

1. Vertical Slicer (VS) as the common entry point for all verticals into the system. The VS dynamically creates and maps the

vertical services onto network slices according to their requirements, and manages their lifecycle. It also translates the vertical and slicing requests into ETSI defined NFV network services (NFV-NS) sent towards the SO. Here a network slice is deployed as a NFV-NS instance.

2. Service Orchestrator (SO). It offers service or resource orchestration and federation, depending on the request coming from the VS. This includes all tasks related with coordinating and offering to the vertical an integrated view of services and resources from multiple administrative domains. Orchestration entails managing end-to-end services or resources that were split into multiple administrative domains based on requirements and availability. Federation entails managing administrative relations at the interface between SOs belonging to different domains and handling abstraction of services and resources.

3. Mobile Transport and Computing Platform (MTP) as the underlying unified transport stratum, responsible for providing the resources required by the NFV-NS orchestrated by the SO. This includes their instantiation over the underlying physical transport network, computing and storage infrastructure. It also may (de)abstract de MTP resources offered to the SO.

The 5G-TRANSFROMER architecture is quite in line with the general Multi Domain Multi Provider reference architecture depicted in Figure 11. Its mapping to the reference architecture is illustrated in the figure below.

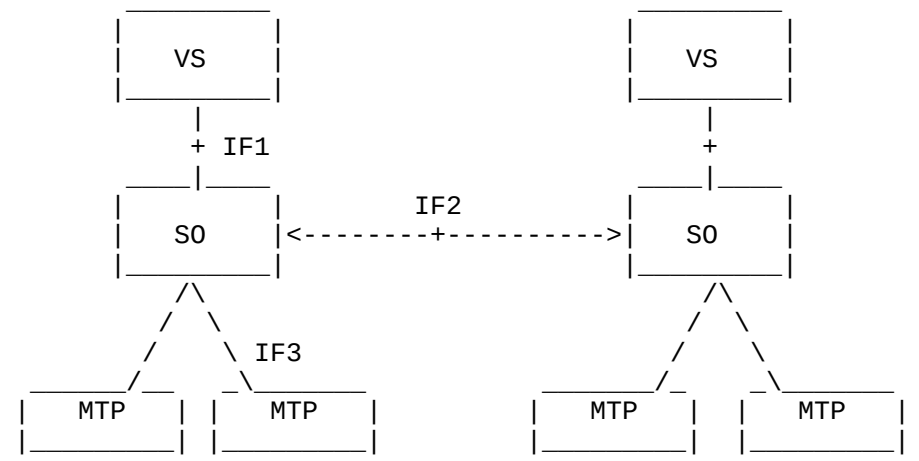


Figure 14: 5G-TRANSFORMER architecture mapped to the reference architecture

The MTP would be mapped to the individual domain orchestrators, which only provides the resource orchestration for the local administrative domain. The role of the SO is the Multi Provider orchestrator (MPO) responsible for multi-domain service or resource orchestration and federation. The operation of the SO follows three main technical steps handled by the three function components of the MPO shown in Figure 14, namely (i) the catalogue and topology management system; (ii) the NFVO (Network Function Virtualization Orchestrator); and the assurance management component.

Correspondingly, the interface between the SO and the VS (So-Vs) is the interface 1 (IF1), through which the VS requests the instantiation and deployment of various network services to support individual vertical service slices. The interface between the SOs (So-So) of different domains is the interface 2 (IF2), enabling multi domain orchestration and federation operations. The interface between the SO and the MTP (So-Mtp) is the interface 3 (IF3). It, on the one hand, provides the SO the updated global view of the underlying infrastructure topology abstraction exposed by the MTP domain orchestrators, while on the other hand it also handles command and control functions to allow the SO request each MTP domain for virtual resource allocation.

In 5G-TRANSFOMER, a set of sub-interfaces have been defined for the So-Mtp, So-So and Vs-So interfaces.

6.3.1. So-Mtp Interface (IF3)

This interface is based on ETSI GS-NFV IFA 005 and ETSI GS-NFV IFA 006 for the request of virtual resource allocation, management and monitoring. Accordingly, the 5G-TRANSFORMER identified the following sub-interfaces at the level of So-Mtp interactions (i.e., IF3-x interfaces regulating MPO-DO interactions).

So-Mtp(-RAM). It provides the Resource Advertisement Management (RAM) functions to allow updates or reporting about virtualized resources and network topologies in the MTP that will accommodate the requested NFVO component network services.

So-Mtp(-RM). It provides the Resource Management (RM) operations over the virtualized resources used for reserving, allocating, updating (in terms of scaling up or down) and terminating (i.e., release) the virtualized resources handled by each MTP and triggered by NFVO component (in Figure 14) to accommodate network services.

So-Mtp(-RMM). It provides the required primitives and parameters for supporting the SO resource monitoring management (RMM)

capability for the purpose of fault management and SLA assurance handled by assurance management component in Figure 14.

In the reference architecture (Fig. 6), the IF3-RC, IF3-RT, IF3-RMon sub-interface are defined for resource control, resource topology and resource monitoring respectively. The IF3-RT, IF3-RC and IF3-RMon sub-interfaces map to So-Mtp(-RAM), So-Mtp(-RM) and So-Mtp(-RMM) sub-interfaces from 5G-TRANSFORMER.

6.3.2. So-So Interface (IF2)

This interface is based ETSI GS-NFV IFA 013 and ETSI GS-NFV IFA 005 for the service and resource federation between the domains. The 5G-TRANSFORMER identified the following sub-interfaces at the level of So-So interactions (i.e., IF2-x interfaces regulating MPO interactions) to provide service and resource federation and enable NSaaS and NFVIaaS provision, respectively, across different administrative domains.

So-So(-LCM), for the operation of NFV network services. The reference point is used to instantiate, terminate, query, update or re-configure network services or receive notifications for federated NFV network services. The SO NFVO-NSO uses this reference point.

So-So(-MON), for the monitoring of network services through queries or subscriptions/notifications about performance metrics, VNF indicators and network service failures. The SO NFVO-NSO uses this reference point.

So-So(-CAT), for the management of Network Service Descriptors (NSDs) flavors together with VNF/VA and MEC Application Packages, including their Application Descriptors (AppDs). This reference point offers primitives for on-boarding, removal, updates, queries and enabling/disabling of descriptors and packages. The SO NFVO-NSO uses this reference point.

Furthermore, resource orchestration related operations are broken up to the following sub-interfaces to reflect resource control, resource topology and resource monitoring respectively.

So-So(-RM), for allocating, configuring, updating and releasing resources. The Resource Management reference point offers operations such as configuration of the resources, configuration of the network paths for connectivity of VNFs. These operations mainly depend of the level of abstraction applied to the actual resources. The SO NFVO-RO uses this reference point.

So-So(-RMM), for monitoring of different resources, computing power, network bandwidth or latency, storage capacity, VMs, MEC hosts provided by the peering administrative domain. The details level depends on the agreed abstraction level. The SO NFVO-RO uses this reference point.

So-So(-RAM), for advertising available resource abstractions to/ from other SOs. It broadcasts available resources or resource abstractions upon capability calculation and periodic updates for near real-time availability of resources. The SO-SO Resource Advertisement uses this reference point.

So-So(-RMM), for monitoring of different resources, computing power, network bandwidth or latency, storage capacity, VMs, MEC hosts provided by the peering administrative domain. The details level depends on the agreed abstraction level. The SO NFVO-RO uses this reference point.

In the reference architecture (Figure 11), the sub-interface IF2-S and IF2-C are defined to perform network service-related operations between MPOs of different administrative domains. The IF2-RC, IF2-RT, IF2-RMon sub-interfaces are defined to regulated interactions between Catalogue and Topology Management components. Their mapping to the sub-interfaces defined in 5G-TRANSFORMER are summarized as follows:

The IF2-S sub-interface maps to So-So(-LCM) and So-So(-MON).

The IF2-C sub-interface maps to So-So(-CAT).

The IF2-RC, IF2-RT, IF2-RMon sub-interfaces map to So-So-RM, So-So-RAM, So-So-RT respectively.

6.3.3. Vs-So Interface (IF1)

This interface is based on ETSI GS-NFV IFA 013 for the VS requesting network services from the SO. Accordingly, the 5G-TRANSFORMER identified the following sub-interfaces at the level of Vs-So interactions (i.e., IF1-x interfaces regulating tenant-MPO interactions).

Vs-So(-LCM). It deals with the NFV network service lifecycle management (LCM) and it is based on the IFA 013 NS Lifecycle Management Interface. It offers primitives to instantiate, terminate, query, update or re-configure network services or receive notifications about their lifecycle.

Vs-So(-MON). It deals with the monitoring (MON) of network services and VNFs through queries or subscriptions and notifications about performance metrics, VNF indicators and network services or VNFs failures. It maps to IF1-S sub-interface of the reference architecture.

Vs-So(-CAT). It deals with the catalogue (CAT) management of Network Service Descriptors (NSDs), VNF packages, including their VNF Descriptors (VNFDs), and Application Packages, including their Application Descriptors (AppDs). It offers primitives for on-boarding, removal, updates, queries and enabling/disabling of descriptors and packages. It maps to IF1-C sub-interface of the reference architecture.

In the reference architecture (Figure 11), the sub-interface IF1-S and IF1-C are defined to build request to perform network service-related operations including requesting the instantiation, update and termination of the requested network services. The IF1-S sub-interface maps to Vs-So(-LCM) and Vs-So(-MON), while the IF1-C sub-interface maps to Vs-So(-CAT) defined in 5G-TRANSFORMER architecture.

7. Multi-domain orchestration and Open Source

Before reviewing current state of the open source projects it should be explicitly mentioned that term "federation" is quite ambiguous and used in multiple contexts across the industry. For example, federation is the approach used at certain software projects to achieve high availability and enable reliable non-interrupted operation and service delivery. One of the distinguishing features of this federation type is that all federated instances are managing the same piece of the infrastructure or resources set. However, this document is focused on another federation type, where multiples independent instances of the orchestration/management software establish certain relationships and expose available resources and capabilities in the particular domain to consumers at another domain. Besides sharing resource details, multi-domain federation requires various management information synchronization, such authentication/ authorization data, run-time policies, connectivity details and so on. This kind of functionality and appropriate implementation approaches at the relevant open source projects are in scope of current section.

At this moment several open source industry projects were formed to develop integrated NFV orchestration platform. The most known of them are ONAP [onap], OSM [osm] and Cloudify [cloudify]. While all these projects have different drivers, motivations, implementation approach and technology stack under the hood, all of them are considering multi-VIM deployment scenario, i.e. all these software

platforms are capable to deploy NFV service over different virtualized infrastructures, like public or private providers. Additionally OSM and Cloudify orchestration platforms have capabilities to manage interconnection among managed VIMs using appropriate plugins or drivers. However, despite the fact that typical Telco/Carrier infrastructure has multiple domains (both technology and administrative), none of these orchestration projects is focused on a service federation use case development.

In the meantime, as an acknowledgement of the challenges, emerged during exploitation of the federation use cases Multisite project emerged under OPNFV umbrella [opnfv]. Considering OpenStack-based VIM deployments spanned across multiple regions as a general use case, this project initially was focusing on a gaps identification in the key OpenStack projects which lacks capabilities for multi-site deployment. During several development phases of this OPNFV project, number of gaps were identified and submitted as a blueprints for the development into the appropriate OpenStack projects. Further several demo scenarios were delivered to trial OpenStack as the open source VIM which is capable to support multisite NFV clouds. While Multisite OPNFV project was focusing on a resource and VIM layer only, there are multiple viable outputs which might be considered during implementation of the federation use cases on the upper layers.

As a summary it can be stated that it is still early days for the technology implemented in a referenced NFV orchestration projects and federation use case in not on a radar for these projects for the moment. However, it is expected that upon maturity of the federation as a viable market use case appropriate feature set in the reviewed projects will be developed.

8. IANA Considerations

N/A.

9. Security Considerations

TBD.

10. 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). This work is also supported by 5G-PPP 5G-TRANSFORMER, a research and innovation action project partially funded by the European Community under the H2020 Program (grant agreement no. 761536). 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.

11. Informative References

[cloudify] "Cloudify", <https://cloudify.co/>.

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

[etsi_nvf_ifa028] "Report on architecture options to support multiple administrative domains, ETSI GR NFV-IFA 028 V3.1.1", January 2018.

[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.

[ngmn_5g_whitepaper] "5G White Paper", February 2015.

[ngmn_slicing] "Description of Network Slicing Concept", January 2016.

[onap] "ONAP project", <https://www.onap.org/>.

[opnfv] "OPNFV Multisite project", <https://wiki.opnfv.org/display/multisite/Multisite>.

[osm] "Open Source MANO project", <https://osm.etsi.org/>.

Authors' Addresses

Carlos J. Bernardos (editor)
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

Josep Mangues-Bafalluy
CTTC
Av. Carl Friecrish Gauss, 7
Castelldefels, EMEA 08860
Spain

Email: josep.mangues@cttc.cat

Xi Li
NEC
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Email: Xi.Li@neclab.eu

Francesco Paolucci
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Phone: +395492124
Email: fr.paolucci@santannapisa.it

Andrea Sgambelluri
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Phone: +395492132
Email: a.sgambelluri@santannapisa.it

Barbara Martini
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Email: barbara.martini@cnit.it

Luca Valcarenghi
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Email: luca.valcarenghi@santannapisa.it

Giada Landi
Nextworks
Via Livornese, 1027
Pisa 56122
Italy

Email: g.landi@nextworks.it

Dmitriy Andrushko
MIRANTIS

Email: dandrushko@mirantis.com

Alain Mourad
InterDigital Europe

Email: Alain.Mourad@InterDigital.com
URI: http://www.InterDigital.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.

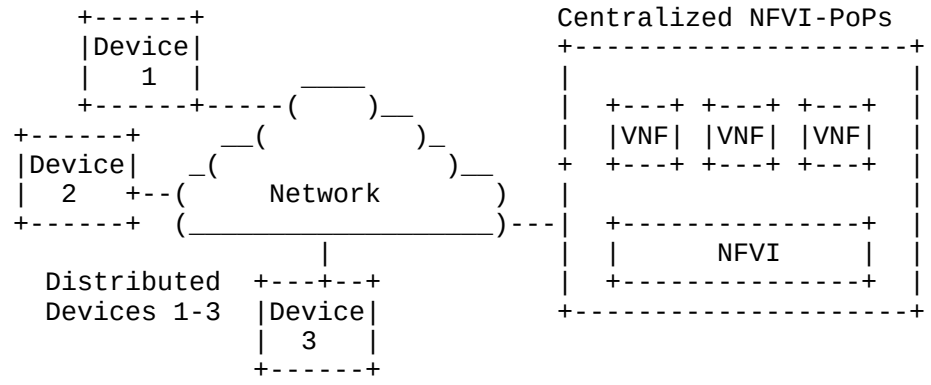


Figure 1

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.

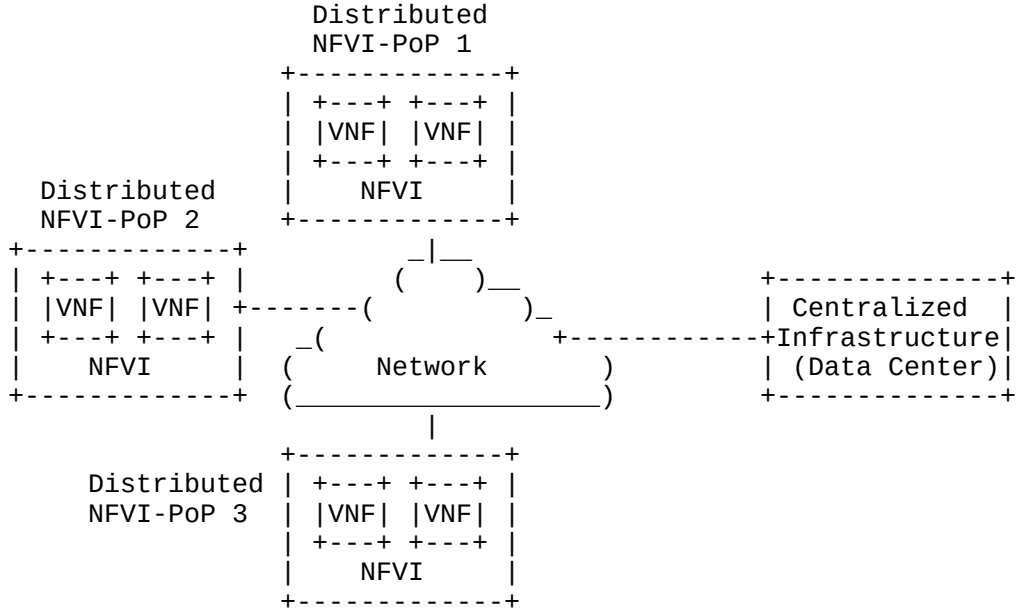


Figure 2

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 R. Gu
Internet-Draft S. Hu
Intended status: Informational China Mobile
Expires: January 3, 2018 July 2, 2017

Control and User Plane Separation Architecture of BNG
draft-gu-nfvrg-cloud-bng-architecture-01

Abstract

This document defines the new architecture of BNG devices with control plane (CP) and user plane (UP) separation. BNG-CP is a user control management component while BNG-UP takes responsibility as the network edge and user policy implementation componenet. Both BNG-CP and BNG-UP are core components for fixed broadband services and deployed separately at different network layer in actual network.

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 separation is to extract and centralize the user management function of multiple BNG devices forming a separate and concentrated CP, while UP takes function as traditional router's control plane 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. C/U separated 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 module or device which can be called BNG-CP (Control Plane). The reserved functions such as router function and forwarding engine can be deployed in the form of BNG User Plane. Thus the Cloud-based BNG architecture is made up of control plane and user plane.

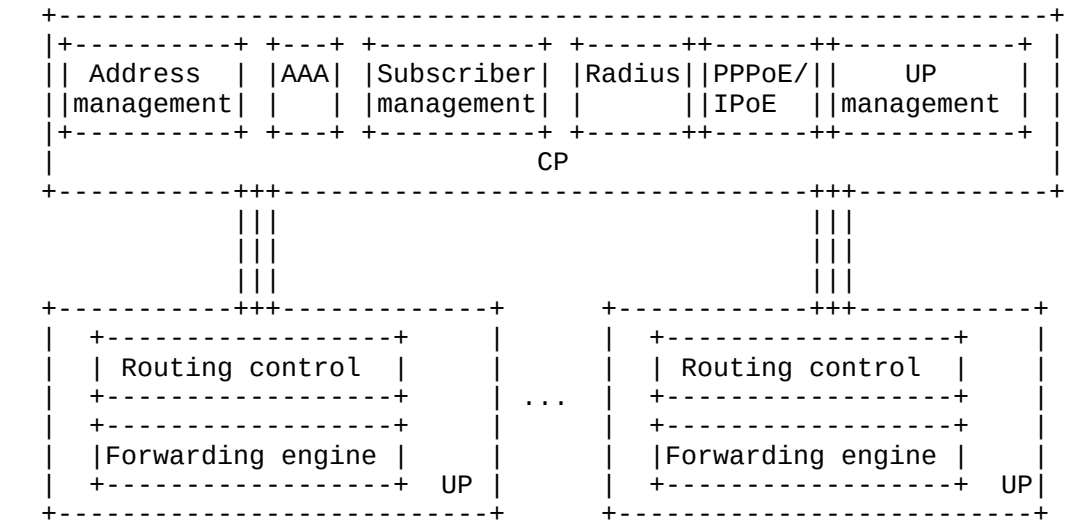


Figure 1: Architecture of C/U Seperation BNG

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

5. C/U separated BNG use case

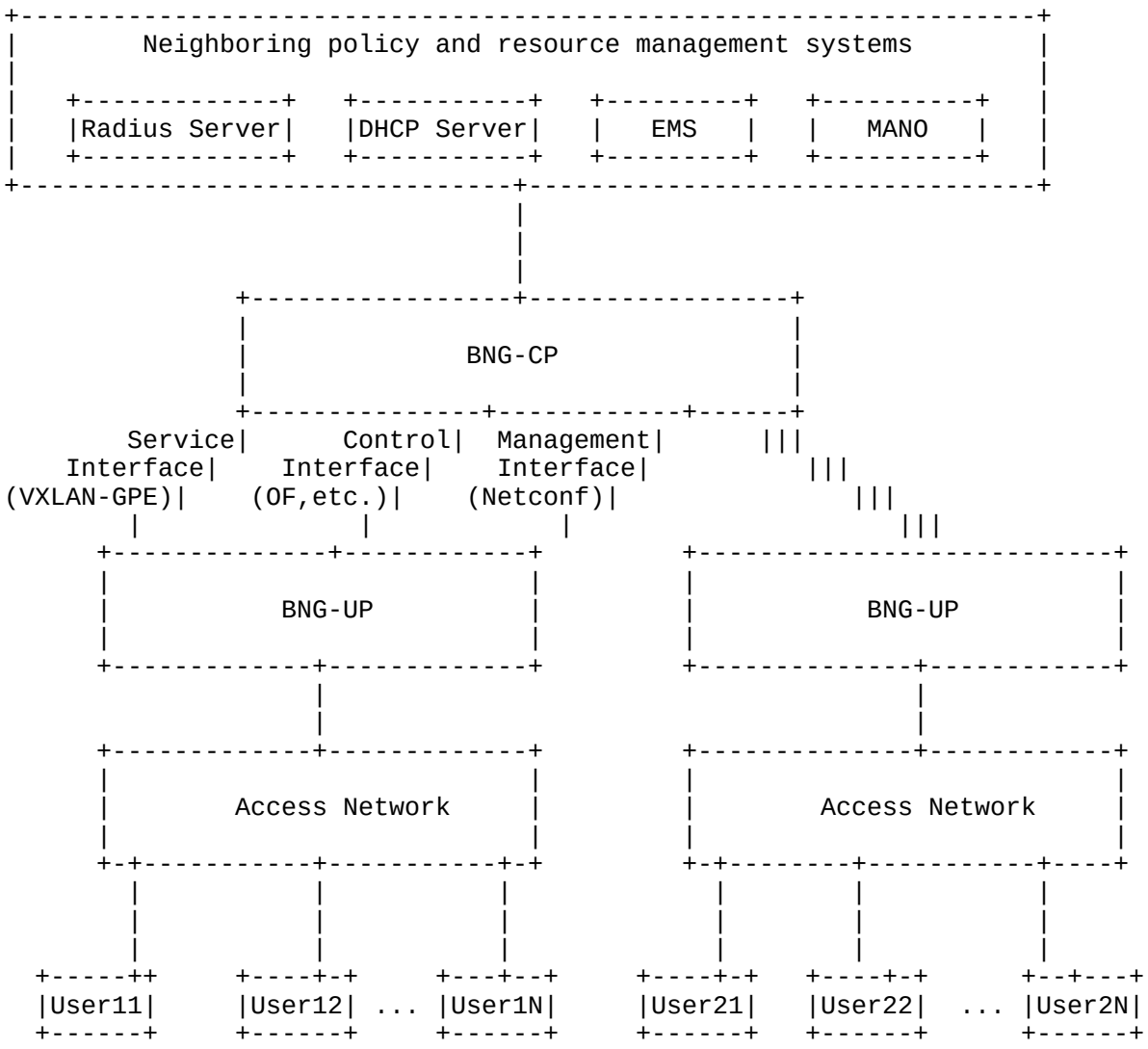


Figure 2: Cloud BNG use case

In the actual deployment, a C/U separated BNG device is composed of CP and UPs. CP is centralized deployed which takes responsiblity of a user control management component managing UP's resources such as the user entry and forwarding policy. And UP is distributed in the bottom acting as a network edge and user policy implementation component.

In order to fulfill a service, Neighboring policy and resource management systems is deployed outside. In the neighboring system, different service systems such as RADIUS server,DHCP server and EMS are included. Besides if BNG-CP is virtualized as a NFV. The NFV infrastructure management system MANO is also included here. BNG-CP has connections with the outside neighboring systems to transmit management traffic.

There are three kinds of interfaces between BNG-CP and BNG-UP according to its traffic catergories: Service Interface, Control Interface, and Management Interface. Service Interface is used to transmit PPPoE/IPoE packets for the authentication. Control Interface is used for setting forwarding entries on UP. Some choose OpenFlow as the protocol. Management Interface is used to carry out basic configurations through NETCONF.

Besides, now we have three related drafts which describes these interfaces in detail. One is VXLAN-GPE extention draft for C/U separated BNG related with Servie Interface in [draft-huang-nov3-vxlan-gpe-extension-for-vbng-00]. One is YANG data model for Management Interface in [draft-hu-opsawg-cu-separation-yang-model-00]. The other is the information model covering Control Interface and Management Interface which makes the abstraction of information in modeling in [draft-wcg-i2rs-cu-separation-info-model-01].

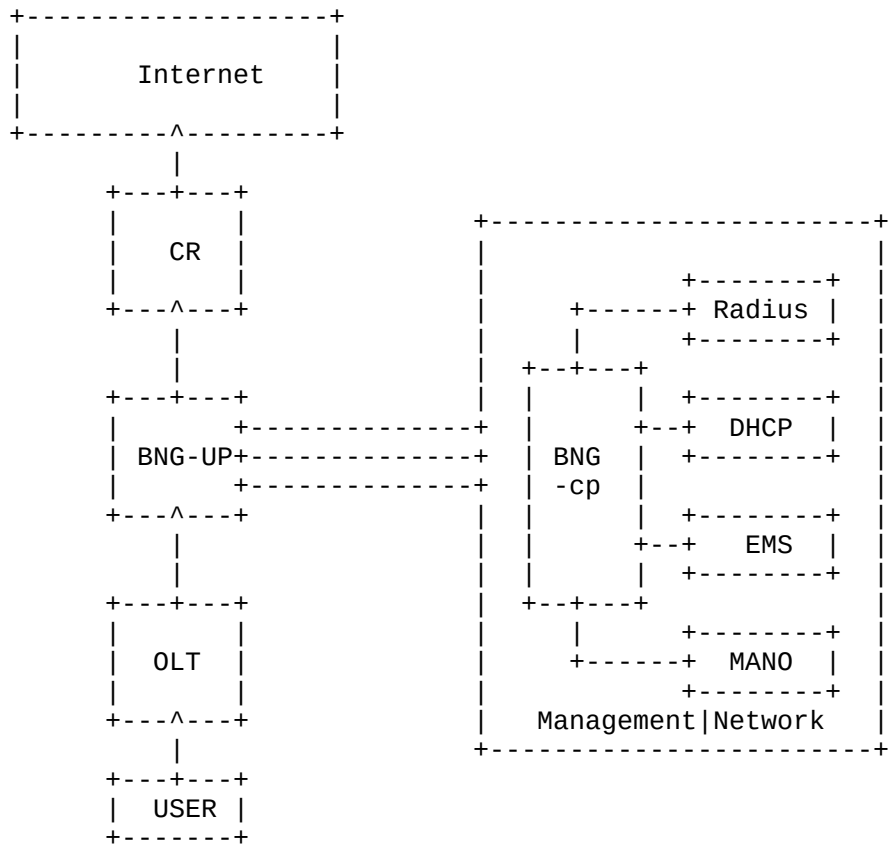


Figure 3: User Dialup process

In the C/U seprated BNG architecture, there are several processes when a home user accesses the Internet.

(1)User dialup packets of PPPoE or IPoE from BNG-UP which will send to BNG-CP from BNG-UP’s Service Interface.

(2)BNG-CP processes the dialup packet. Confirming with the outside neighboring systems in the management network, BNG-CP makes the decision to permit or deny of the dial through certification.

(3)After that, BNG-CP tells UP to do the responding forwarding actions with related policies.

(4)If the user is certificated and permitted, the UP forwards the traffic into the Internet with related policies such as limited

bandwidth, etc. Otherwise, the user is denied to access the Internet.

6. Advantages of C/U seperated BNG

Due to the bandnew C/U separated BNG architecture, there are a lot of brilliant advantages.

(1)Resources can be central controlled and balanced

Centralized control plane takes the responsibility of control and management. Thus it has the overall view of resources and can distribute the resources as required.

(2)Device can be more efficient in extension

Control plane and user plane can be extended separately according to different situations such as the session overload and extremely high throughput.

(3)Management can be much easier as the BNG-CP is the only one facing to the outside system such as EMS, DHCP server, Radius and so on.

(4)BNG-CP can be virtualized as a VNF with its management of MANO.

(5)BNG-UP can be a virtual machine or physical device as demand.

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 1, 2018

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

February 28, 2018

Network Virtualization Research Challenges
draft-irtf-nfvrg-gaps-network-virtualization-09

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 and engineering challenges in network virtualization including guaranteeing quality-of-service, performance improvement, supporting multiple domains, network slicing, service composition, device virtualization, privacy and security, separation of control concerns, network function placement and testing. In addition, some proposals are made for new activities in IETF/IRTF that could address some of these challenges. This document is a product of the Network Function Virtualization Research Group (NFVRG).

Copyright Notice

Table of Contents

1. Introduction and scope

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 moving 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 and edge servers/services 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 and engineering challenges in network virtualization that need to be addressed in order to achieve these goals, spanning from pure research and engineering/standards space. 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.

This document represents the consensus of the NFV Research Group. It has been reviewed by the Research Group members active in the specific areas of work covered by the document.

2. Terminology

The following terms used in this document are defined by the ETSI Network Function Virtualization (NVF) Industrial Study Group (ISG) [etsi_gs_nfv_003], the ONF [onf_tr_521] and the IETF [RFC7426] [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.

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 infrastructure operator's 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

This section briefly describes some basic background technologies, as well as other standards developing organizations and open source initiatives working on network virtualization or related topics.

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. The ETSI NFV is one of the predominant NFV reference framework and architectural footprints [nfv_sota_research_challenges]. 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.

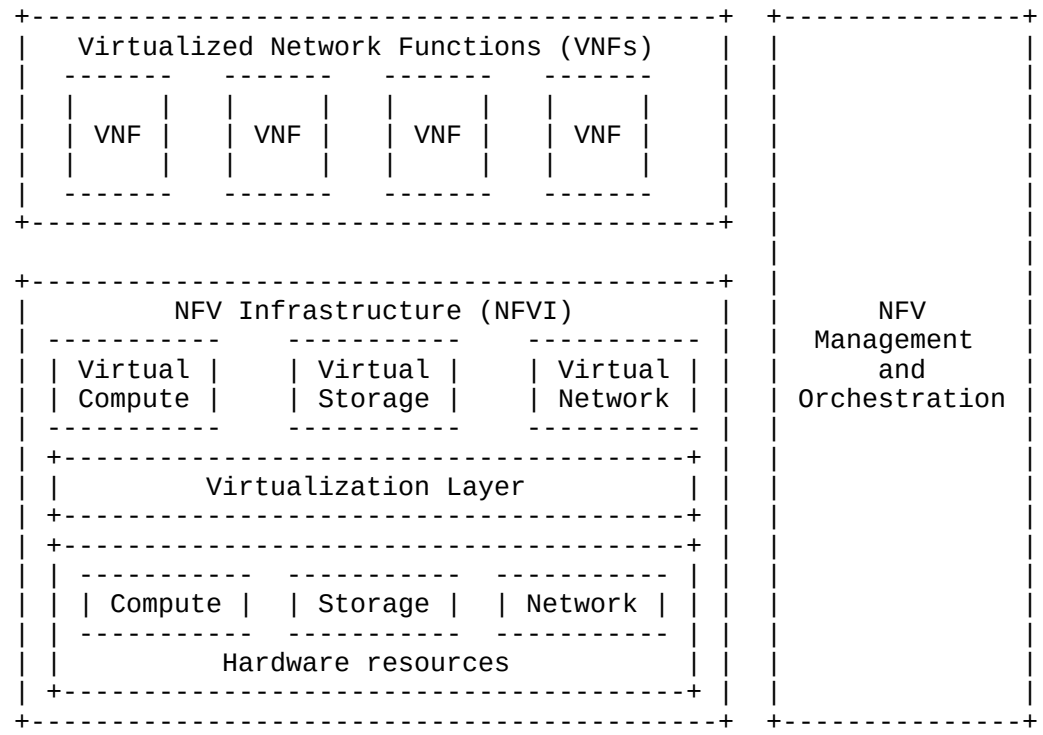


Figure 1: ETSI NFV framework

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).

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

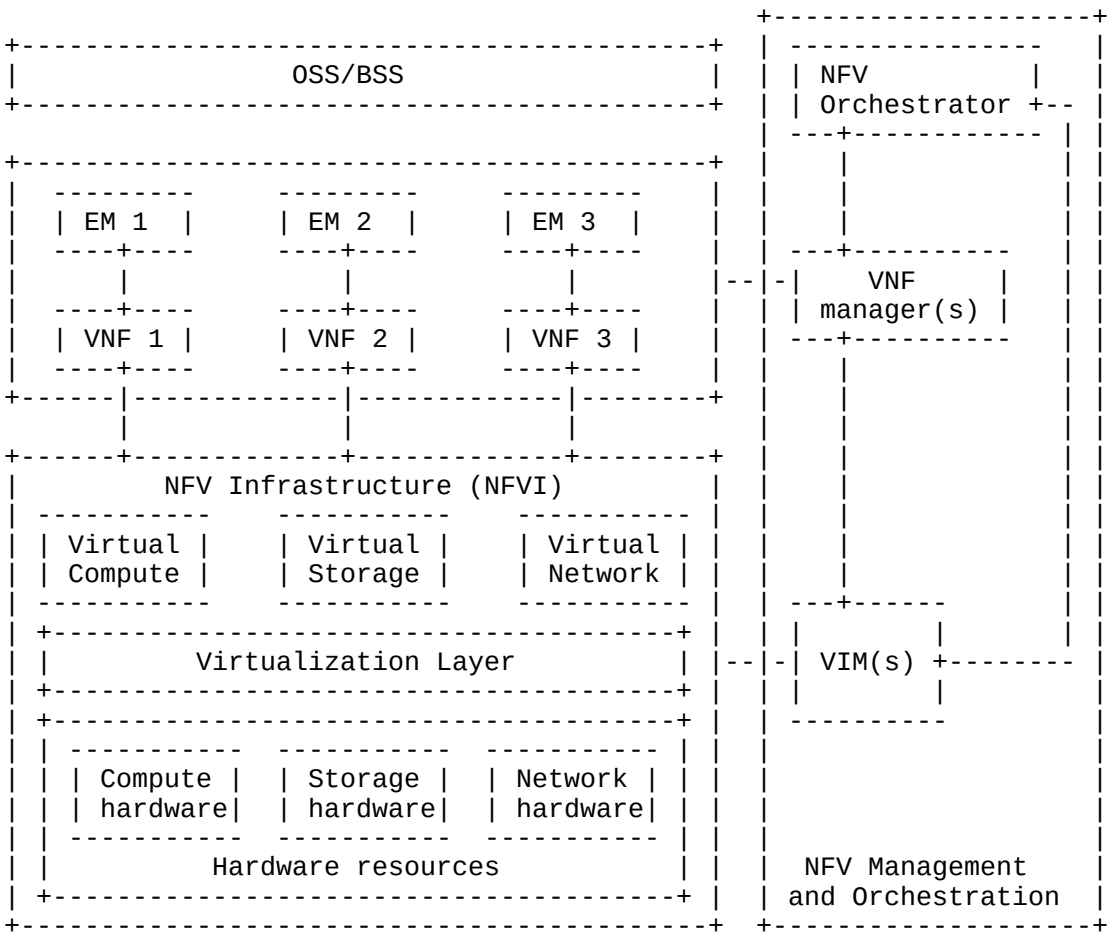


Figure 2: ETSI NFV reference architecture

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 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 to manage their network, i.e., an operator can easily change its policies on how traffic is distributed throughout the network.

One of the most well known protocols for the SDN control plane between the central controller and the networking elements 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.

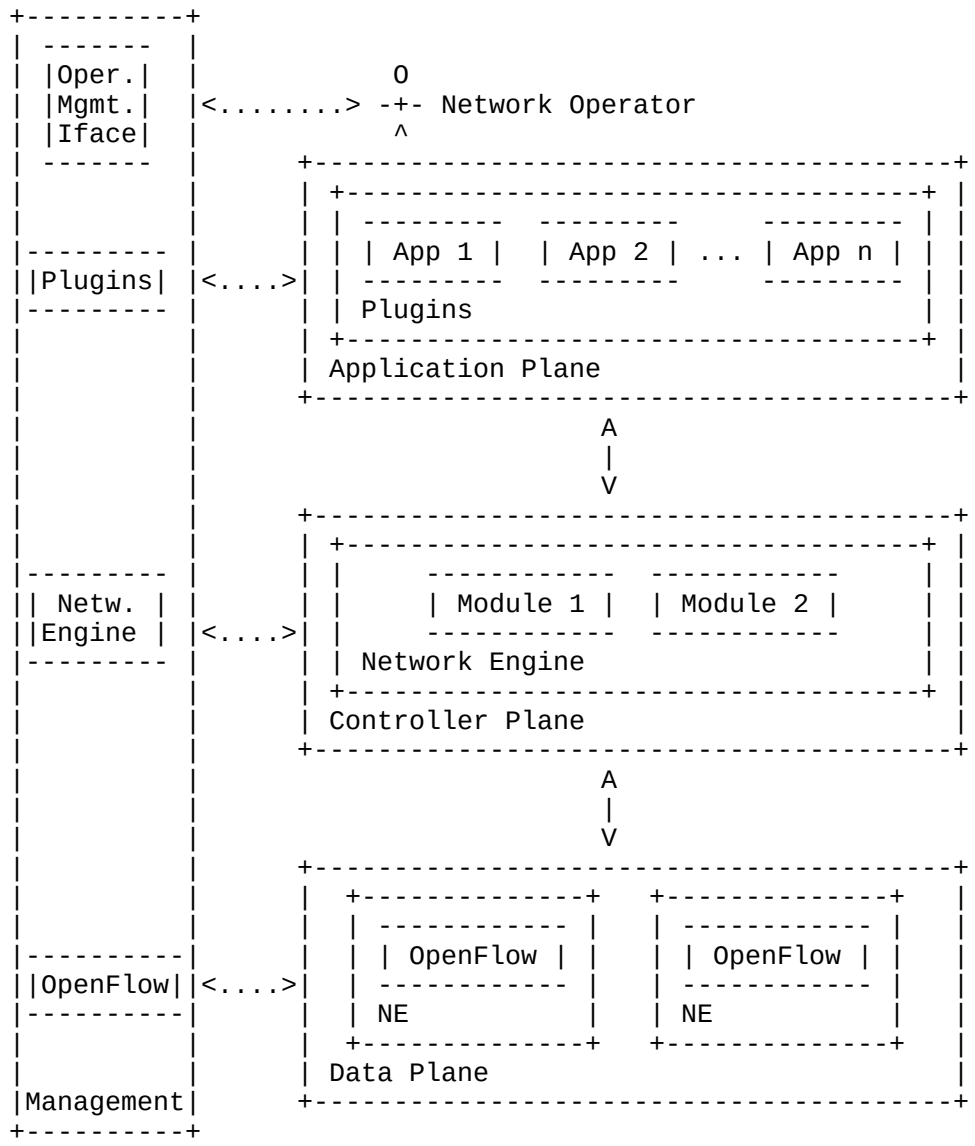


Figure 3: High level SDN ONF architecture

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 underlying 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 these interactions, the Management plane exposes several functions to network operators which can easily and quickly configure and tune the network at each layer.

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

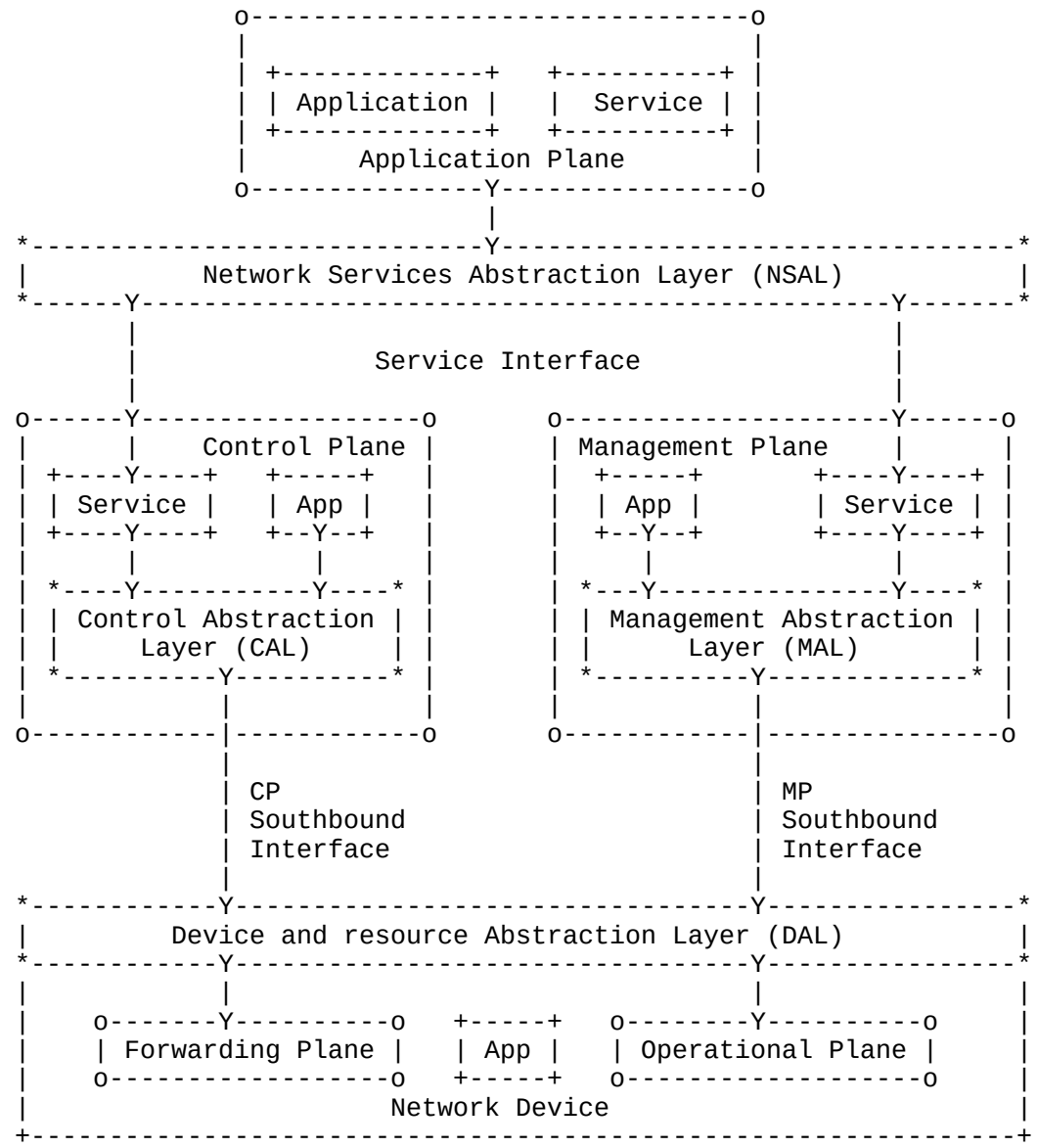


Figure 4: SDN Layer Architecture

While SDN is often directly associated to OpenFlow, this is just one (relevant) example of a southbound protocol between the central controller and the network entities. Other relevant examples of protocols in the SDN family are NETCONF [RFC6241], RESTCONF [RFC8040] and ForCES [RFC5810].

3.3. ITU-T functional architecture of SDN

The Telecommunication standardization sector of the International Telecommunication Union (ITU) -- the ITU-T -- has also looked into SDN architectures, defining a slightly modified one from what other SDOs have done. ITU-T provides in the recommendation ITU-T Y.3302 [itu-t-y.3302] a functional architecture of SDN with descriptions of functional components and reference points. The described functional architecture is intended to be used as an enabler for further studies on other aspects such as protocols and security as well as being used to customize SDN in support of appropriate use cases (e.g., cloud computing, mobile networks). This recommendation is based on ITU-T Y.3300 [itu-t-y.3300] and ITU-T Y.3301 [itu-t-y.3301]. While the first describes the framework of SDN (including definitions, objectives, high-level capabilities, requirements and the high-level architecture of SDN), the second describes more detailed requirements.

Figure 5 shows the SDN functional architecture defined by the ITU-T. It is a layered architecture composed of the SDN application layer (SDN-AL), the SDN control layer (SDN-CL) and the SDN resource layer (SDN-RL). It also has multi-layer management functions (MMF), which provides functionalities for managing the functionalities of SDN layers, i.e., SDN-AL, SDN-CL and SDN-RL. MMF interacts with these layers using MMFA, MMFC, and MMFR reference points.

The SDN-AL enables a service-aware behavior of the underlying network in a programmatic manner. The SDN-CL provides programmable means to control the behavior of SDN-RL resources (such as data transport and processing), following requests received from the SDN-AL according to MMF policies. The SDN-RL is where the physical or virtual network elements perform transport and/or processing of data packets according to SDN-CL decisions.

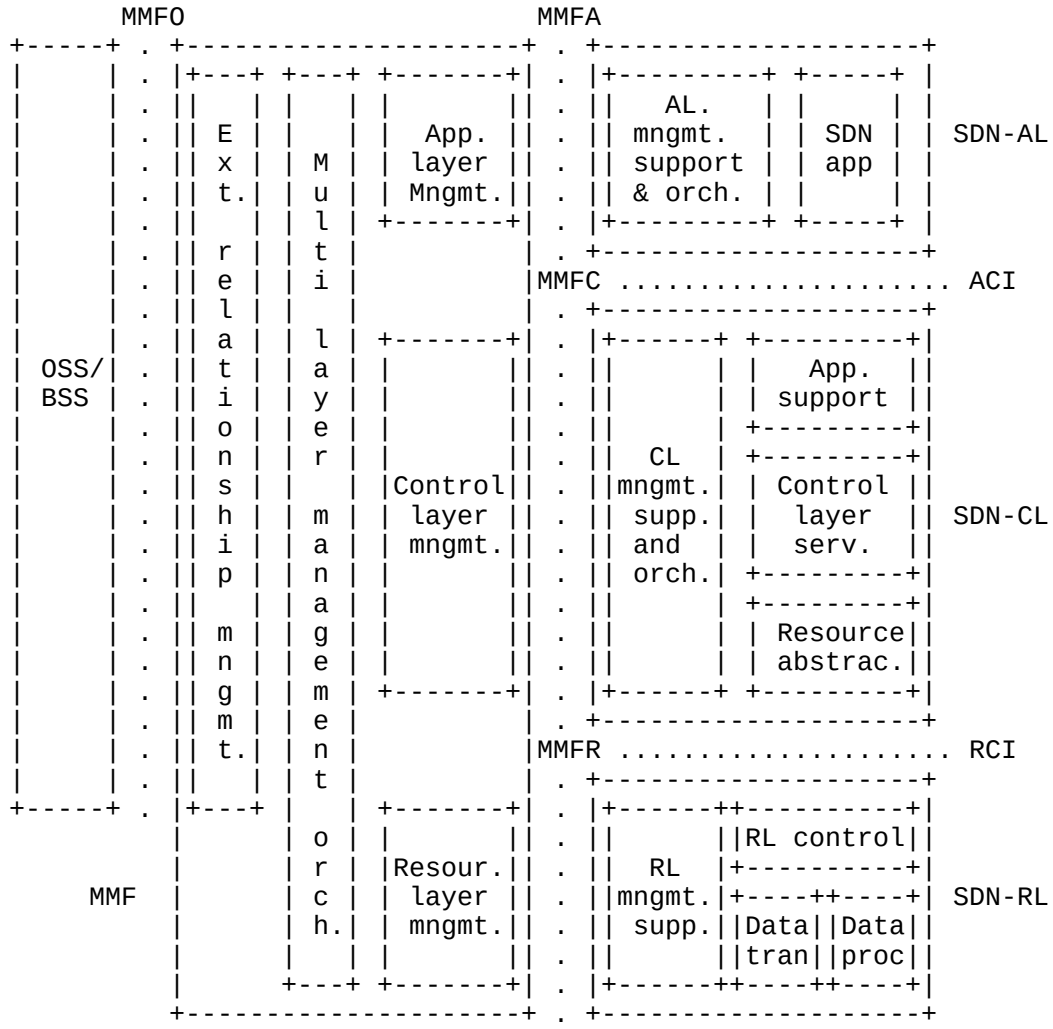


Figure 5: ITU-T SDN functional architecture

3.4. Multi-access Edge Computing

Multi-access Edge Computing (MEC) -- formerly known as 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.

3.5. 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 SDN principles, thereby lowering the barriers to new network technologies, to new network operators, and to new service providers.

3.6. Distributed Management Task Force

The DMTF (https://www.dmtf.org/) 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.

3.7. Open Source initiatives

The Open Source community is especially active in the area of network virtualization and orchestration. 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.

- Kubernetes. Kubernetes is an open-source system for automating deployment, scaling and management of containerized applications. Kubernetes can schedule and run application containers on clusters of physical or virtual machines. Kubernetes allows: (i) Scale on the fly, (ii) Limit hardware usage to required resources only, (iii) Load balancing Monitoring, and (iv) Efficient lifecycle management.

- OpenDayLight. OpenDaylight (ODL) is a highly available, modular, extensible and scalable 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 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 (i) increase performance and power efficiency, (ii) improve reliability, availability, and serviceability, and (iii) 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.

- ONAP. ONAP (Open Network Automation Platform) is an open source software platform that delivers capabilities for the design, creation, orchestration, monitoring, and life cycle management of: (i) Virtual Network Functions (VNFs), (ii) The carrier-scale Software Defined Networks (SDNs) that contain them, and (iii) Higher-level services that combine the above. ONAP (derived from the AT&T's ECOMP) provides for automatic, policy-driven interaction of these functions and services in a dynamic, real-time cloud environment.

- SONA. SONA (Simplified Overlay Network Architecture) is an extension to ONOS to have a almost full SDN network control in OpenStack for virtual tenant network provisioning. Basically, SONA is an SDN-based network virtualization solution for cloud DC.

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

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 moving communication services from specific hardware in the operators' core to server farms scattered in datacenters (i.e. compute and storage virtualization). In addition, the networks interconnecting the functions that compose a network service are fundamentally affected in the way they route, process and control traffic (i.e. network virtualization).

4.2. Guaranteeing quality-of-service

Achieving a given quality-of-service in an NFV environment with virtualized and distributed computing, storage and networking functions is more challenging than providing the equivalent in

discrete non-virtualized components. 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]. Again, the comparison point is against a router or forwarder built on optimized hardware. 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 has 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), Single Root I/O Virtualization (SR-IOV), Non-Uniform Memory Access (NUMA), Data Plane Development Kit (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, UDP) 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, at the time of writing 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, [RFC8172] 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 [RFC8172], 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 looking 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, guaranteeing 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. Implementing functionality in SW over "commodity" infrastructure should make it 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 that are 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 datacenter"), 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 datacenter" 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 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 the overall 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.

Some additional examples of research topics that could enable energy savings are [nfv_sota_research_challenges]:

- Energy aware scaling (e.g., reductions in CPU speeds and partially turning off some hardware com- ponents to meet a given energy consumption target.
- Energy-aware function placement.
- Scheduling and chaining algorithms, for example adapting the network topology and operating parameters to minimize the operation cost (e.g., tracking energy costs to identify the cheapest prices).

Note that it is also important to analyze the trade-off between energy efficiency and network performance.

4.3.2. Improved link usage

The use of NFV and SDN technologies can help improve link usage. SDN has already shown that it can greatly increase average link utilization (e.g., Google example [google_sdn_wan]). NFV adds more complexity (e.g., due to service function chaining / VNF forwarding graphs) 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 at 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 [multi-domain_5GEx].

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] [multi-domain_5GEx].

Another side of the multi-domain problem is the integration/ harmonization of different management domains. A key example comes from Multi-access Edge Computing, which, according to ETSI, comes with its own MANO system, and would require to be integrated if interconnected to a generic NFV system.

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 selective combinations of radio access networks and core network components will have to be combined into a given network slice to address the specific requirements of each use case.

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 time exceeding by a factor of thousands that of smart phones or some other devices that will have almost continuous control and data communications. Hence, it is envisioned that a customized network slice will have to be stitched together from virtual resources or sub-slices to meet these requirements.

The actual definition of network slice from an IP infrastructure viewpoint is currently undergoing intense debate [I-D.geng-coms-problem-statement] [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 the new business model enables performance that meets the application needs (e.g., broadcasting updates to many sensors with satellite broadcasting capabilities). However, more work needs to be done to define the basic architectural approach of how network

slices will be defined and formed. For example, is it mostly a matter of defining the appropriate network models (e.g. YANG) to stitch the network slice from existing components. Or do end-to-end timing, synchronization and other low level requirements mean that more fundamental research has to be done.

4.5.1. Virtual Network Operators

The widespread use/discussion/practice of system and network virtualization technologies has led 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 for the reliability of the virtual resources but there are several situations in which a 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 give such overlay controllers the possibility to 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 [RFC8030].

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 adaptation 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 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 the 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 (see Section 4.5).

The specific nature of the IoT ecosystem, particularly reflected in the Machine-to-Machine (M2M) communications, leads to the creation of new and highly distributed systems which demand location-based network and computing services. A 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. In summary, the research gap is exact localization of VNFs at far network edge infrastructure which is highly distributed and dynamic.

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 necessarily 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) [sfc_challenges], 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 means 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 accomplishing aforementioned goals of "load balancing and improve reliability".

As described in [dynamic_chaining], different algorithms can be used to enable dynamic service composition that optimizes a QoS-based utility function (e.g., minimizing the latency per-application traffic flows) for a given composition plan. Such algorithms can consider the computation capabilities and load status of resources

executing the VNF instances, either deduced through estimations from historical usage data or collected through real-time monitoring (i.e., context-aware selection). For this reason, selections should include references to dynamic information on the status of the service instance and its constituent elements, i.e., monitoring information related to individual VNF instances and links connecting them as well as derived monitoring information at the chain level (e.g., end-to-end delay). At runtime, if one or more VNF instances are no more available or QoS degrades below a given threshold, the service selection task can be rerun to perform service substitution.

There are different research directions that relate to the previous point. For example, the use of Integer Linear Programming (ILP) techniques can be explored to optimize the management of diverse traffic flows. Deep machine learning can also be applied to optimize service chains using information parameters such as some of the ones mentioned above. Newer scheduling paradigms, like co-flows, can also be used.

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 for service function chaining 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 (e.g., 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, for example, being explored 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 service providers 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 a certain level of security in an NFV system. 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 focuses 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.

On the other hand, privacy refers to concerns about the control of personal data and the decision of what to reveal to whom. 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 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 hardware (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 (exactly) one 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 involved, making security policy coordination 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? Alternatively, it can be investigated how to develop a mechanism that avoid such a security policy coordination, this making the system more robust.

On a positive note, NFV may 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. This holds true in particular if each data center is protected separately via firewalls, DMZs and other network protection techniques.

SDN can also be used to help improve security by facilitating the operation of existing protocols, such as Authentication, Authorization and Accounting (AAA). The management of AAA infrastructures, namely the management of AAA routing and the establishment of security associations between AAA entities, can be performed using SDN, as analyzed in [I-D.marin-sdnrg-sdn-aaa-mng].

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 advantage 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 malfunctioning or interference concerns. These ideas are considered in [etsi_gs_nfv_eve005] and [I-D.irtf-sdnrg-layered-sdn]

4.10. Network Function placement

Network function placement is a problem in any kind of network telecommunications infrastructure. Moreover, the increased degree of freedom added by network virtualization makes this problem even more important, and also harder to tackle. Deciding where to place virtual network functions is a resource allocation problem which needs to (or may) take into consideration quite a few aspects: resiliency, (anti-)affinity, security, privacy, energy efficiency, etc.

When several functions are chained (typical scenario), placement algorithms become more complex and important (as described in Section 4.6). While there has been research on the topic [nfv_piecing] [dynamic_placement][vnf-p], this still remains an open challenges that requires more attention. Multi-domain also adds another component of complexity to this problem that has to be considered.

4.11. Testing

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

1. Changes in methodology.

2. New functionality.

3. Opportunities.

4.11.1. 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.

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, 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, 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 about 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 are critical. In a multi-tenant environment, 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.

Test beds for any type of testing for an NFV-based system will be largely similar to previously used test architectures. The methods are impacted by virtualization, as described above, but the design of

test beds are similar as in the past. There are two main new considerations:

- Since networking is based on software, which has lead to greater automation in deployment, the test system should also be deployable with the rest of the system in order to fully automate the system. This is especially relevant in a DevOps environment supported by a CI/CD tool chain (see Section 4.11.3 below).

- In any performance test bed, the test system should not share the same resources as the System Under Test (SUT). While multi-tenenacy is a reality in virtualization, having the test system share resources with the SUT will impact the measured results in a performance test bed. The test system should be deployed on a separate platform in order to not to impact the resources available to the SUT.

4.11.2. 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.

4.11.3. 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 Continuous Integration and Continuous Deployment in the NFV environment. The CI/CD 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, but can also be 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 and IRTF 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	SFC WG, NFVRG (energy driven orchestration)
3-Multiple Domains	NFVRG (multi-domain orchestration)
4-Network Slicing	NVO3 WG, NETSLICES bar BoF (multi-tenancy support)
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 (separation between transport control and services)
9-Testing	NFVRG (testing of scaling)
10-Function placement	NFVRG, SFC WG (VNF placement algorithms and protocols)

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

6. NFVRG focus areas

Table 2 correlates the currently identified NFVRG topics of interests/focus areas to the open network virtualization research areas enumerated in this document. This can help the NFVRG in identifying and prioritizing research topics. The current list of NFVRG focus points is the following:

- Re-architecting functions, including aspects such as new architectural and design patterns (e.g., containerization, statelessness, serverless, control/data plane separation), SDN integration, and proposals on programmability.

- New management frameworks, considering aspects related to new OAM mechanisms (e.g., configuration control, hybrid descriptors) and lightweight MANO proposals.

- Techniques to guarantee low latency, resource isolation, and other dataplane features, including hardware acceleration, functional offloading to dataplane elements (including NICs), and related approaches.

- Measurement and benchmarking, addressing both internal measurements and external applications.

NFVRG Focus Point	Open Research Area
1-Re-architecting functions	- Performance improvem.
	- Network Slicing
	- Guaranteeing QoS
	- Security
	- End-user device virt.
	- Separation of control
2-New management frameworks	- Multiple Domains
	- Service Composition
	- End-user device virt.
3-Low latency, resource isolation, etc	- Performance improvem.
	- Separation of control
4-Measurement and benchmarking	- Guaranteeing QoS
	- Testing

Table 2: Mapping of NFVRG Focus Points 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, Saumya Dikshit, Fabio Giust, Evangelos Haleplidis, Angeles Vazquez-Castro, Barbara Martini, Jose Saldana and Gino Carrozzo 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 authors want to also thank Dave Oran and Michael Welzl for their very detailed IRSG reviews.

The work of Carlos J. Bernardos and Luis M. Contreras is partially supported by the H2020 5GEx (Grant Agreement no. 671636) and 5G-TRANSFORMER (Grant Agreement no. 761536) projects.

10. Informative References

[dynamic_chaining]
Martini, B. and F. Paganelli, "A Service-Oriented Approach for Dynamic Chaining of Virtual Network Functions over Multi-Provider Software-Defined Networks", Future Internet vol. 8, no. 2, June 2016.

[dynamic_placement]
Clayman, S., Maini, E., and A. Galis, "The dynamic placement of virtual network functions", 2014 IEEE Network Operations and Management Symposium (NOMS) pp. 1-9, May 2014.

[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 GS NFV-PER 001 V1.1.2, "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 GS NFV-SEC 001 V1.1.1, "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_3]
"Network Functions Virtualisation (NFV). White Paper 3", October 2014.

[google_sdn_wan]
Sushant Jain et al., "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., Vaishnavi, I., and R. Szabo, "Multi-domain Network Virtualization", draft-bernardos-nfvrg-multidomain-03 (work in progress), September 2017.

[I-D.defoy-netslices-3gpp-network-slicing]
Foy, X. and A. Rahman, "Network Slicing - 3GPP Use Case", draft-defoy-netslices-3gpp-network-slicing-02 (work in progress), October 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.geng-coms-problem-statement]
67, 4., Wang, L., Slawomir, S., Qiang, L., Matsushima, S., Galis, A., and L. Contreras, "Problem Statement of Supervised Heterogeneous Network Slicing", draft-geng-coms-problem-statement-01 (work in progress), October 2017.

[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.marin-sdnrg-sdn-aaa-mng]
Lopez, R. and G. Lopez-Millan, "Software-Defined Networking (SDN)-based AAA Infrastructures Management", draft-marin-sdnrg-sdn-aaa-mng-00 (work in progress), November 2015.

[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]
Torre, P., "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>.

[itu-t-y.3300]
ITU-T, "Y.3300: Framework of software-defined networking", ITU-T Recommendation Y.3300 (06/14), June 2014, <http://www.itu.int/rec/T-REC-Y.3300-201406-I/en>.

[itu-t-y.3301]
ITU-T, "Y.3301: Functional requirements of software-defined networking", ITU-T Recommendation Y.3301 (09/16), September 2016, <http://www.itu.int/rec/T-REC-Y.3301-201609-I/en>.

[itu-t-y.3302]
ITU-T, "Y.3302: Functional architecture of software-defined networking", ITU-T Recommendation Y.3302 (01/17), January 2017, <http://www.itu.int/rec/T-REC-Y.3302-201701-I/en>.

[multi-domain_5GEx]
Bernardos, C., Geroe, B., Di Girolamo, M., Kern, A., Martini, B., and I. Vaishnavi, "5GEx: Realizing a Europe wide Multi-domain Framework for Software Defined Infrastructures", Transactions on Emerging Telecommunications Technologies vol. 27, no. 9, pp. 1271-1280, September 2016.

[nfv_piecing]
Luizelli, M., Bays, L., and L. Buriol, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions", 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) pp. 98-106, May 2015.

[nfv_sota_research_challenges]
Mijumbi, R., Serrat, J., Gorricho, J-L., Bouten, N., De Turck, F., and R. Boutaba, "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, "Recommended Practice for Network Reference Model and Functional Description of IEEE 802 Access Network", Draft 1.0 , December 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]
Lopez, D., "OpenMANO: The Dataplane Ready Open Source NFV MANO Stack", March 2015, <https://www.ietf.org/proceedings/92/slides/slides-92-nfvrg-7.pdf>.

[RFC5810] Doria, A., Ed., Hadi Salim, J., Ed., Haas, R., Ed., Khosravi, H., Ed., Wang, W., Ed., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, DOI 10.17487/RFC5810, March 2010, <https://www.rfc-editor.org/info/rfc5810>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8030] Thomson, M., Damaggio, E., and B. Raymor, Ed., "Generic Event Delivery Using HTTP Push", RFC 8030, DOI 10.17487/RFC8030, December 2016, <https://www.rfc-editor.org/info/rfc8030>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8172] Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", RFC 8172, DOI 10.17487/RFC8172, July 2017, <https://www.rfc-editor.org/info/rfc8172>.

[sfc_challenges] Medhat, A., Taleb, T., Elmangoush, A., Carella, G., Covaci, S., and T. Magedanz, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges", IEEE Communications Magazine vol. 55, no. 2, pp. 216-223, February 2017.

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

[vnf-p] Moens, H. and Filip De Turck, "VNF-P: A model for efficient placement of virtualized network functions", 10th International Conference on Network and Service Management (CNSM) and Workshop pp. 418-423, 2014.

[vnf_benchmarking] Rosa, R., Rothenberg, C., and R. Szabo, "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.

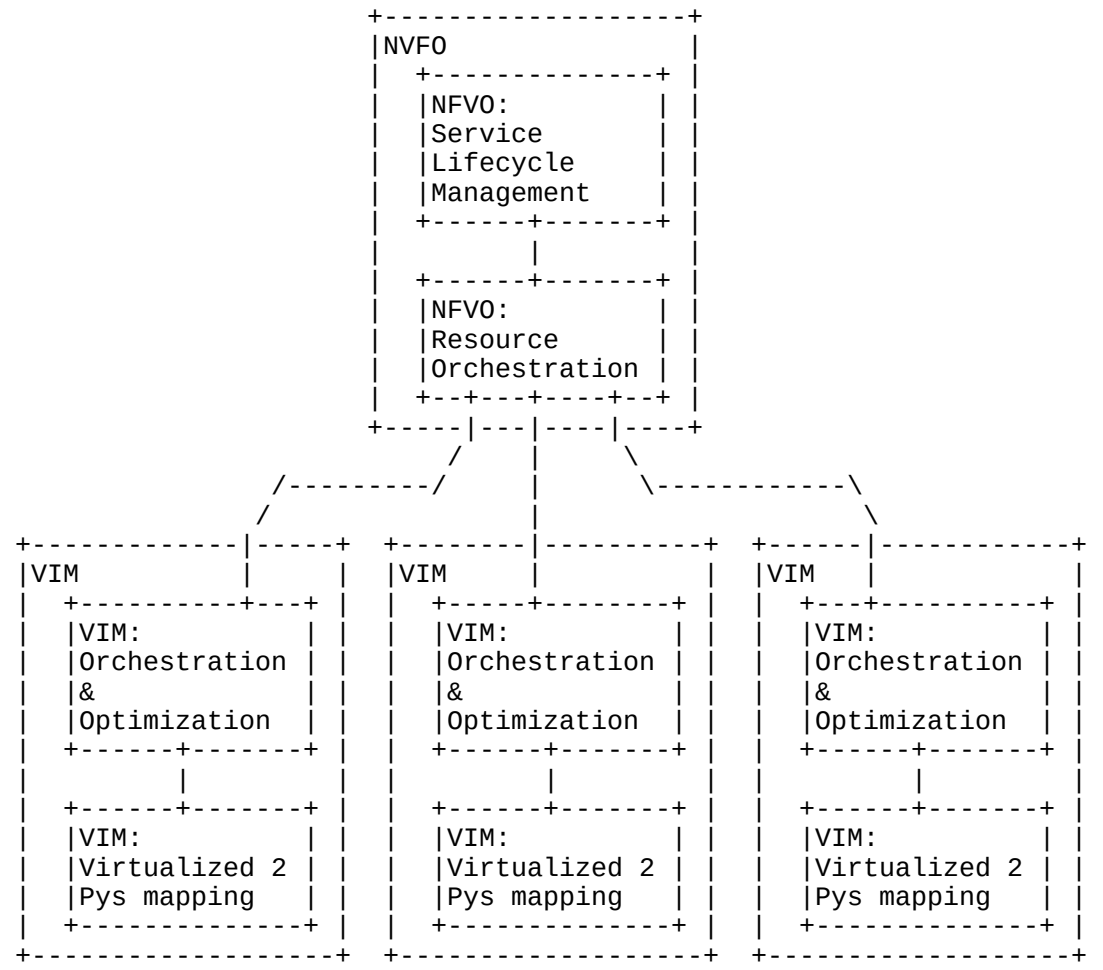


Figure 1: Functional decomposition of the NFVO and the VIM according to the ETSI MANO

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.

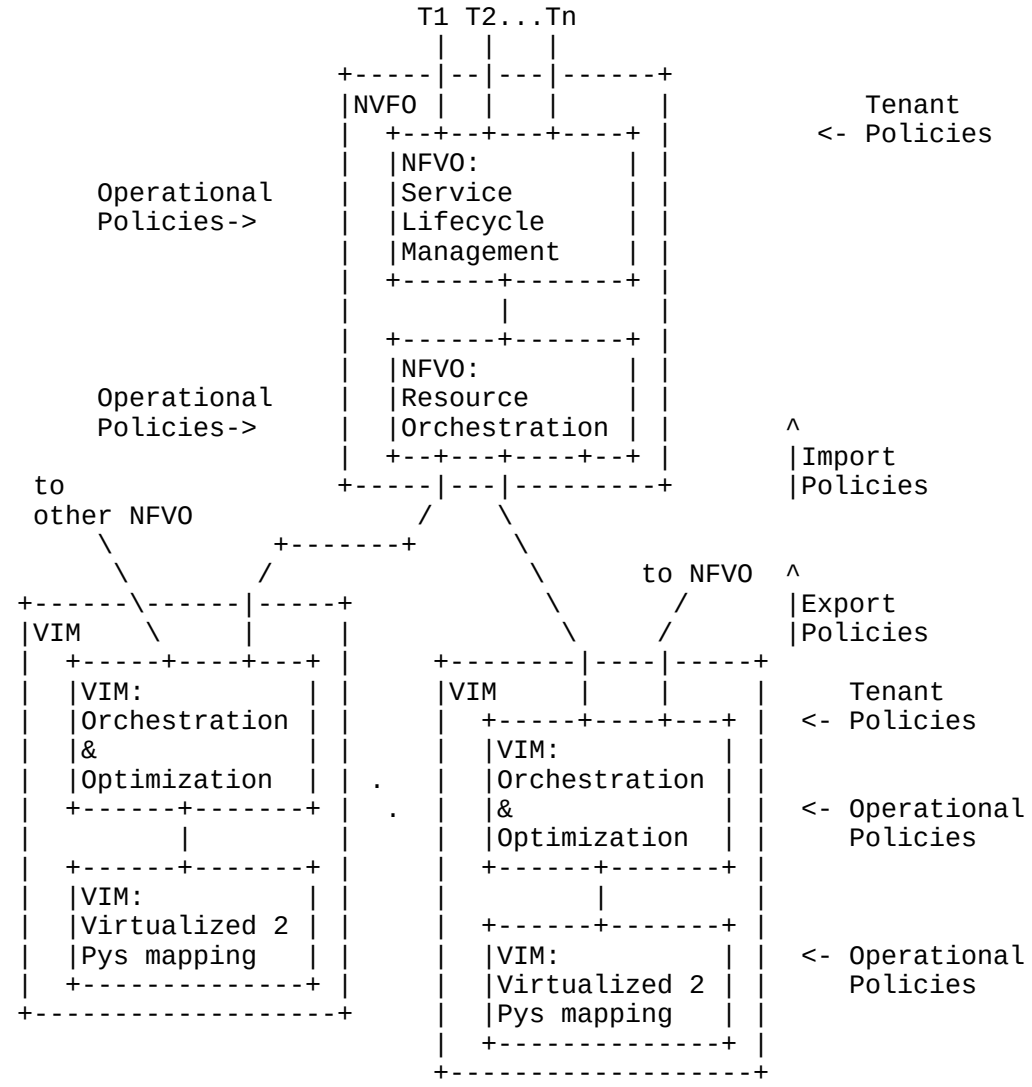


Figure 2: Policies within the MANO framework

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.

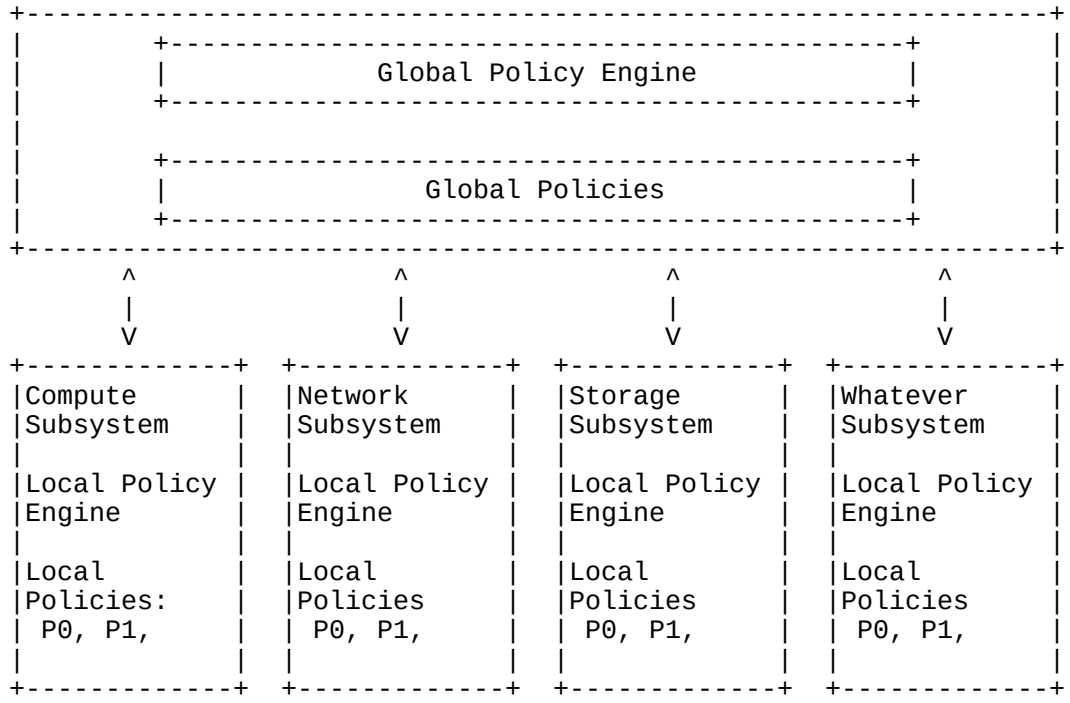


Figure 3: Global versus Local Policy Engines

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.

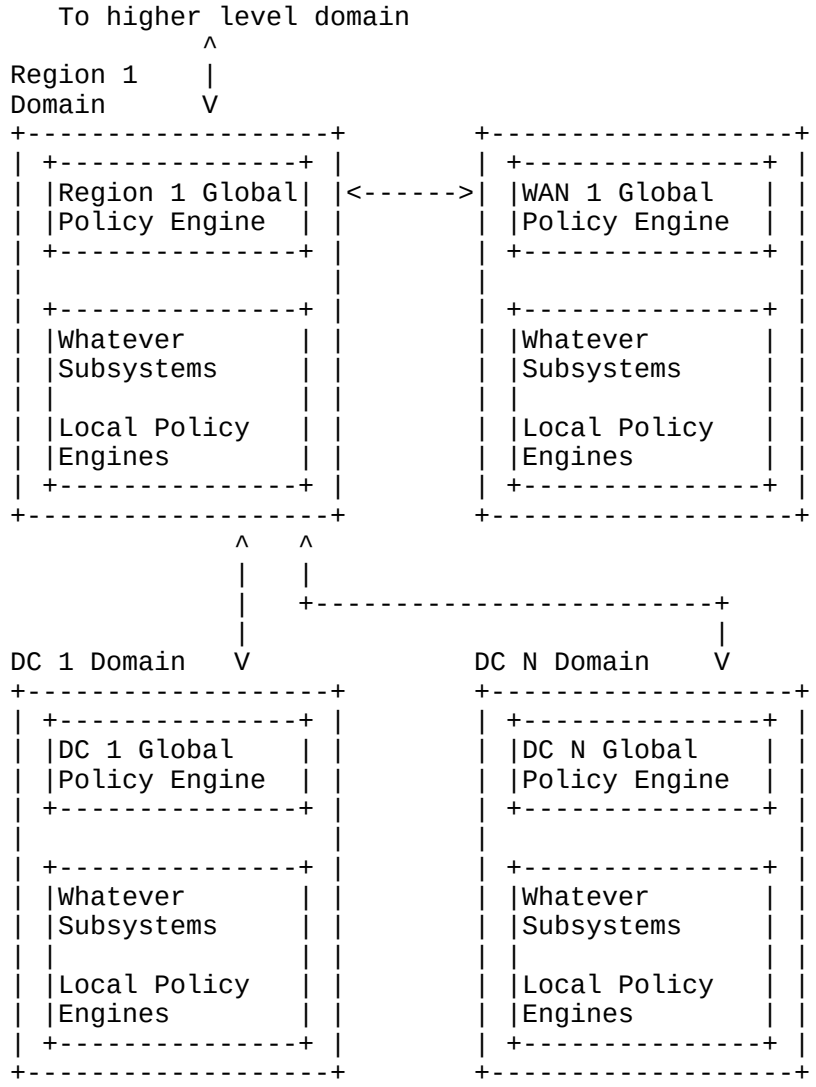


Figure 4: A Hierarchical Policy Framework

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

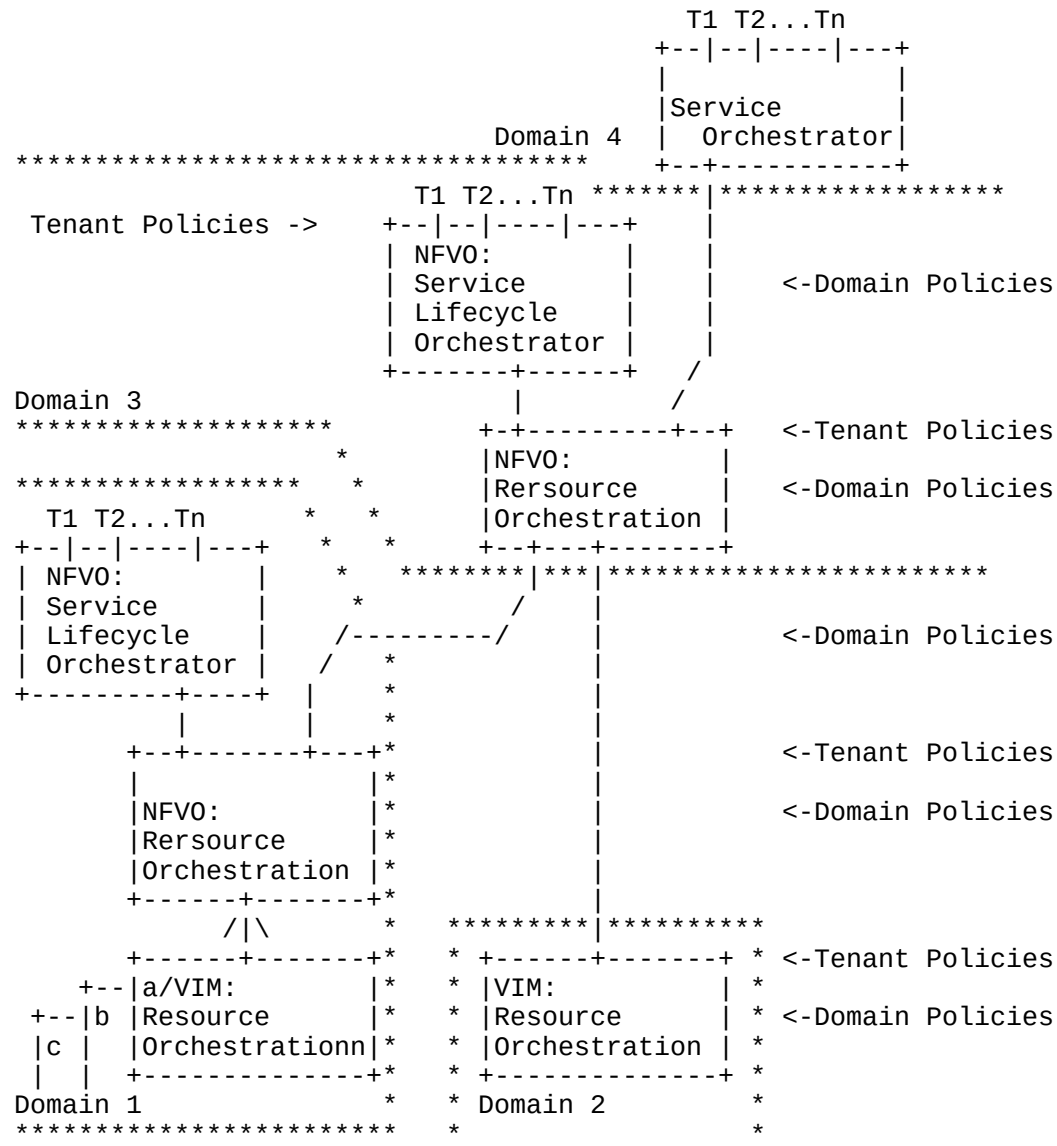


Figure 5: Policies in a Hierarchical Orchestration View

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.

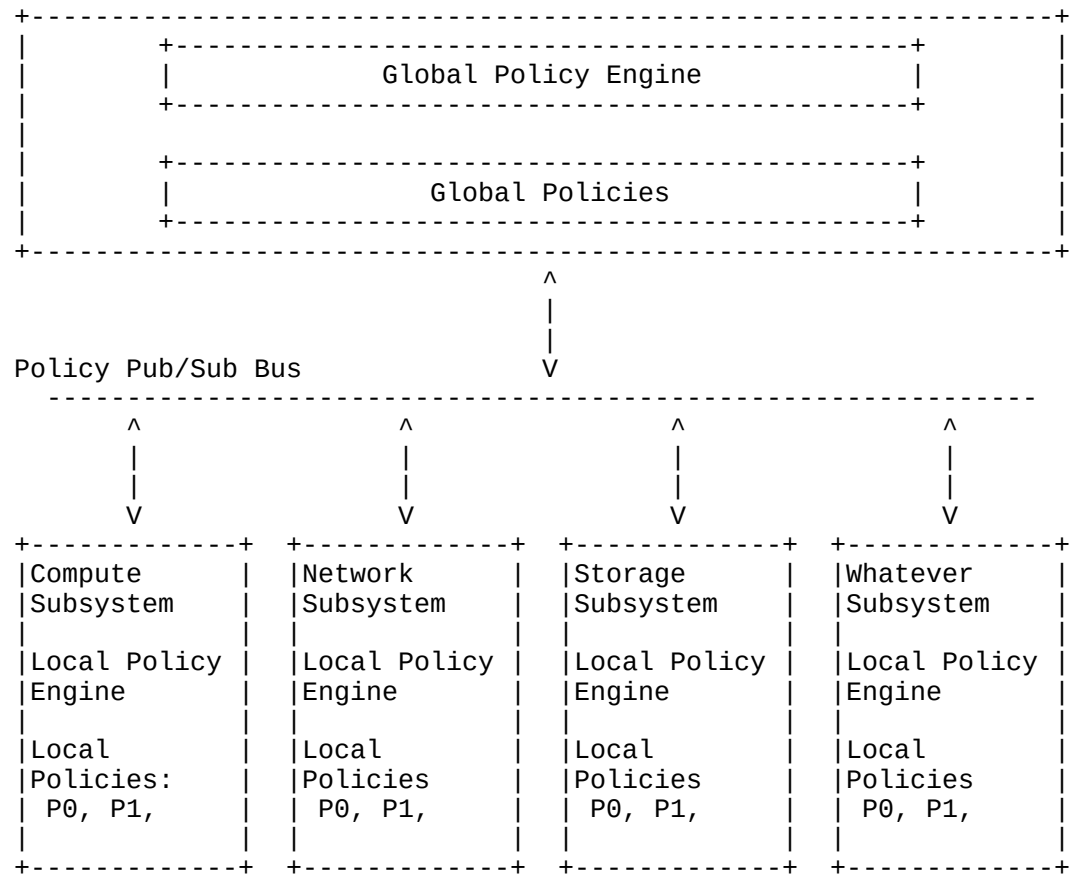


Figure 6: A Policy Pub/Sub Bus

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.

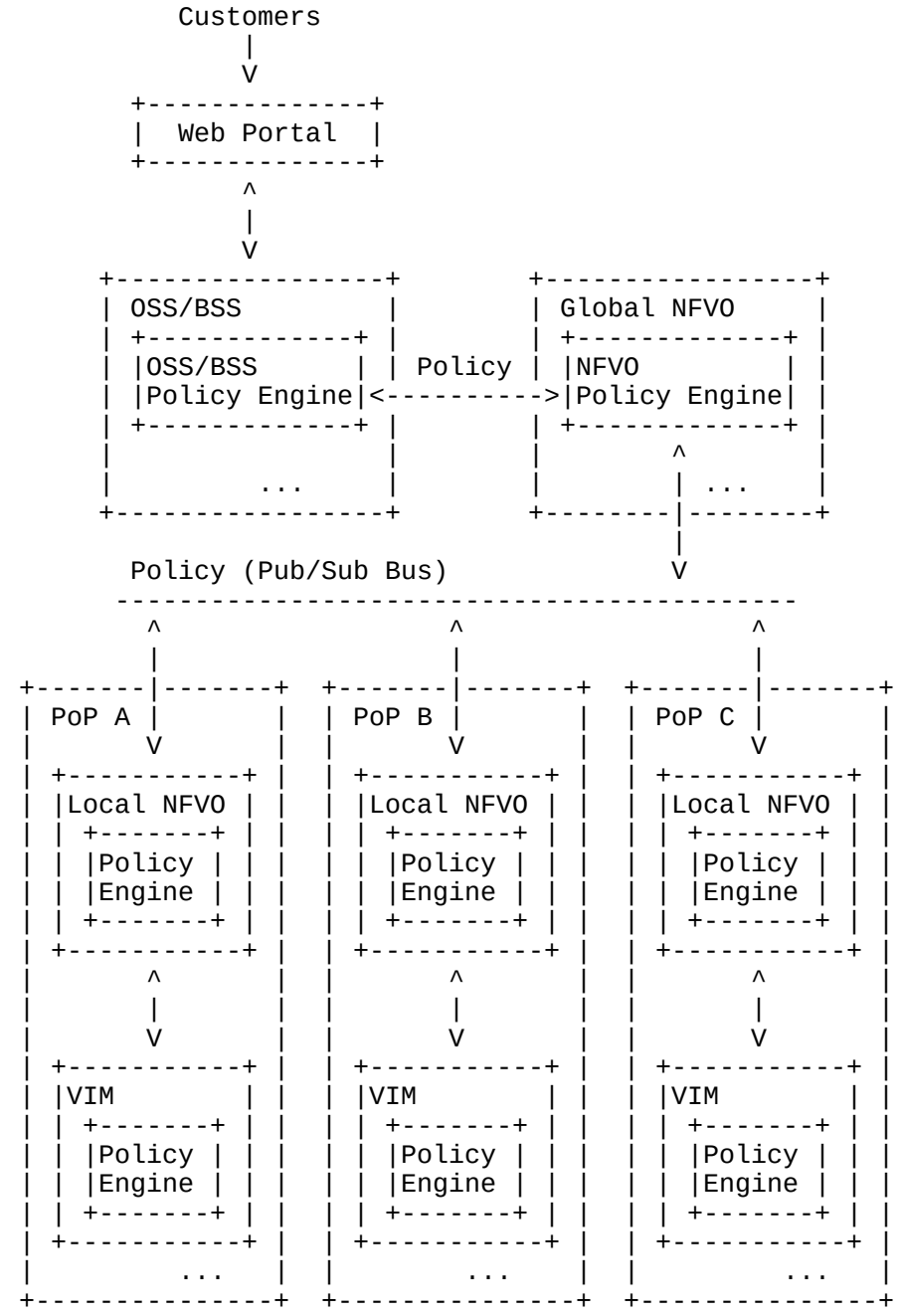


Figure 7: Pub/sub bus in the hierarchical MANO framework

5.6. Policy Intent Statement versus Subsystem Actions and Configurations

Content to be merged

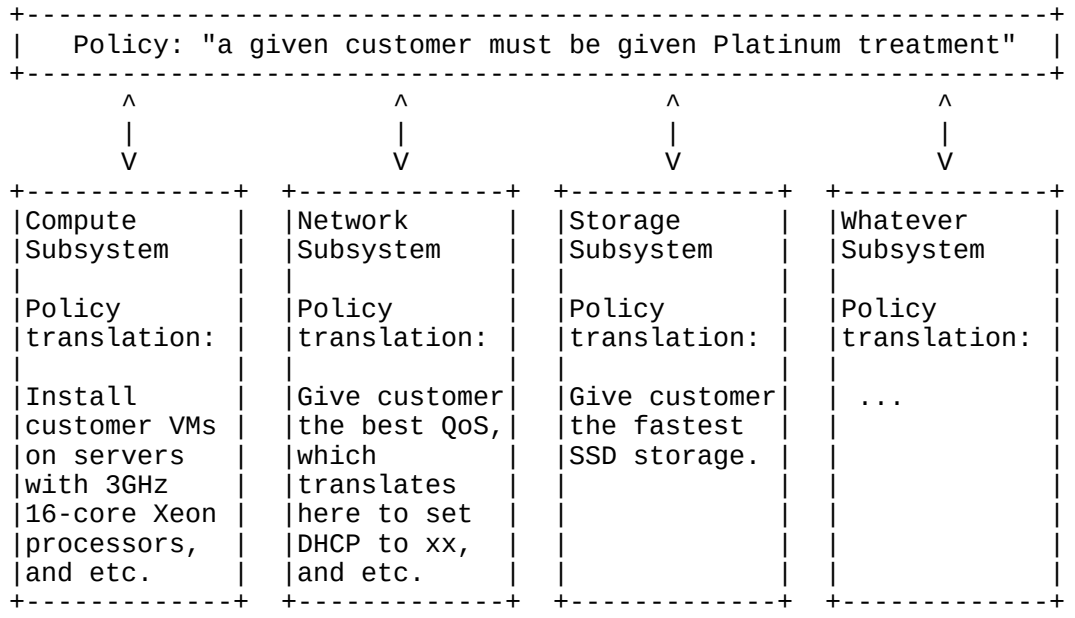


Figure 8: Example of Subsystem Translations of Policy Actions

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

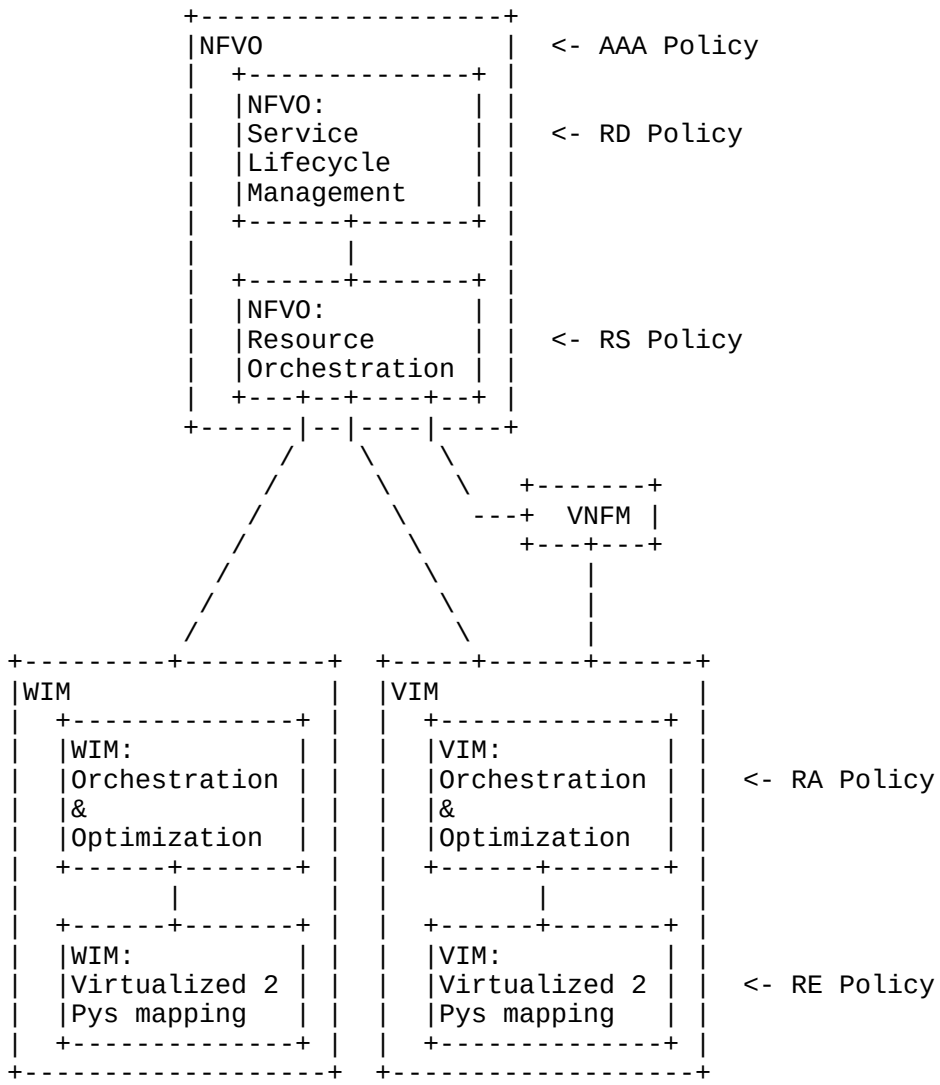


Figure 9: Operational policies for resource management

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

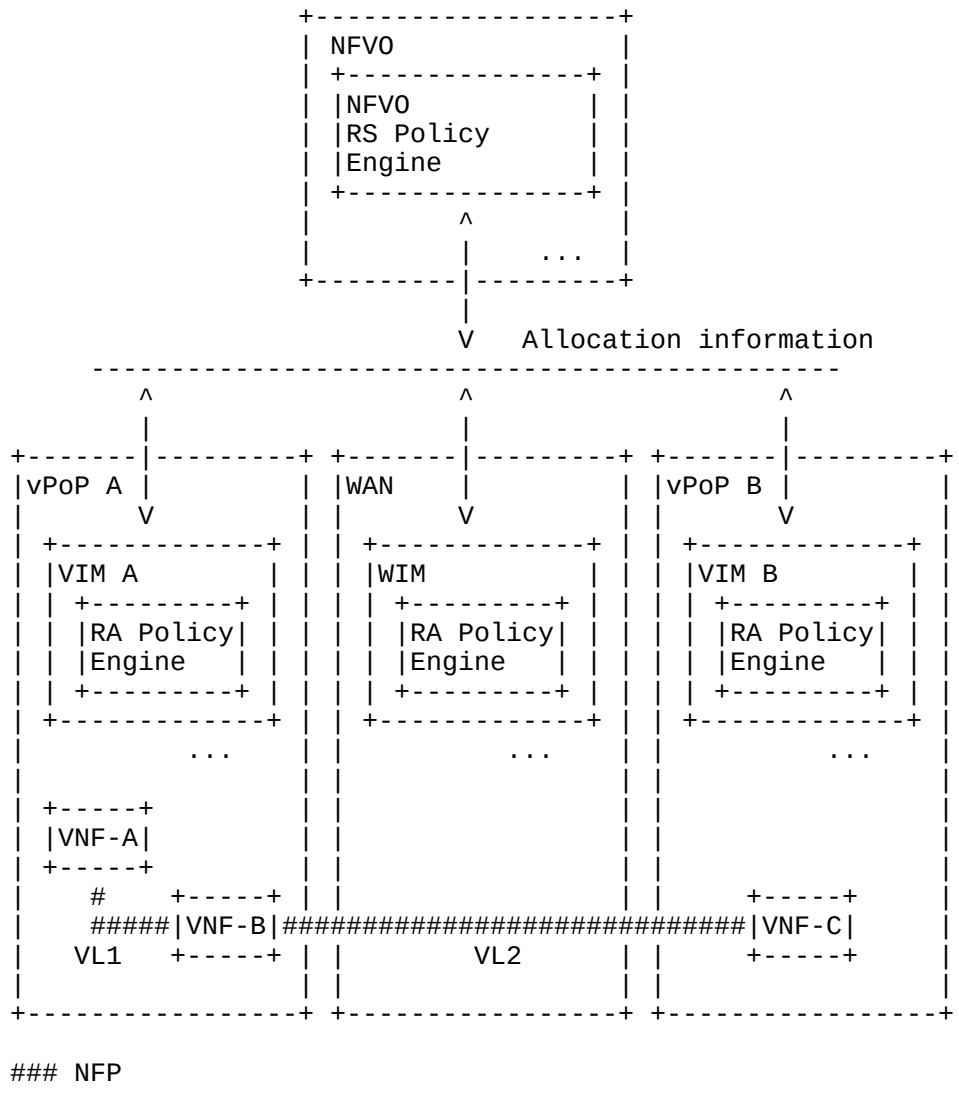


Figure 10: Policy-based VNF-FG Management

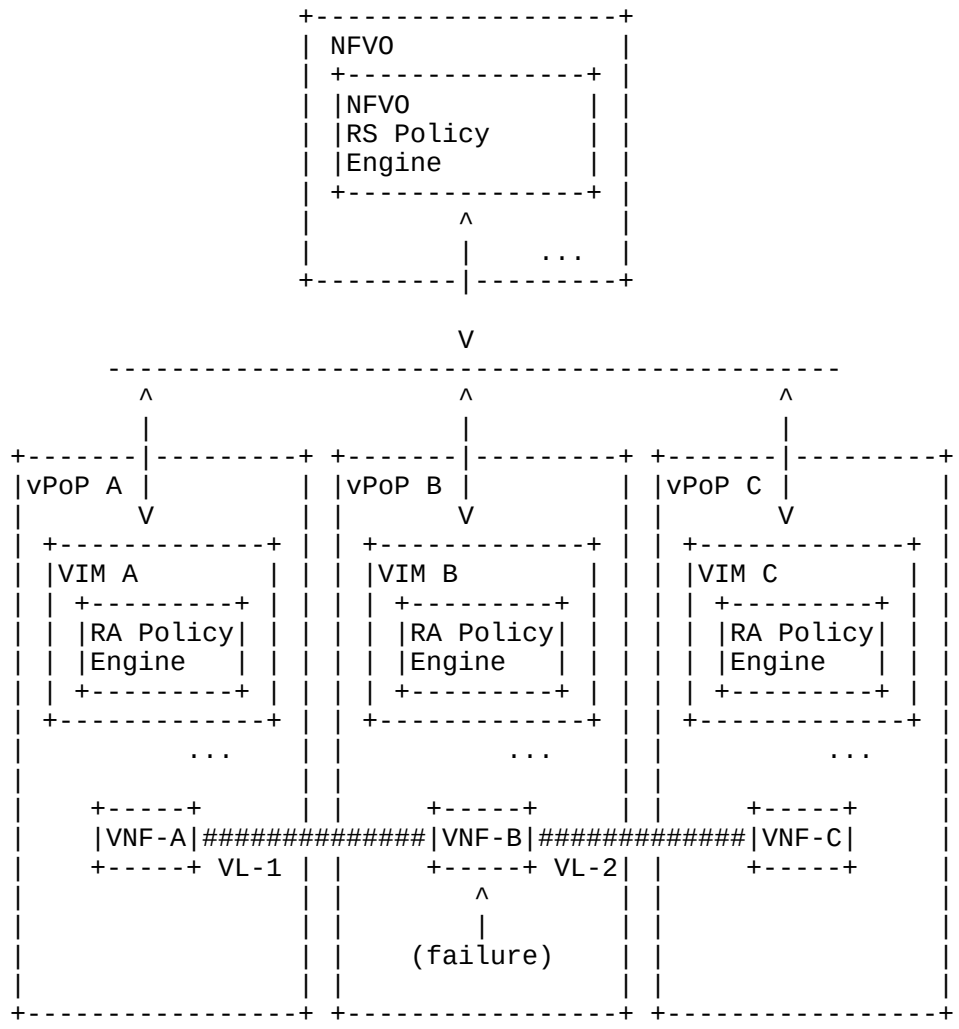
Another subsystem example for the policy framework is VNF-FG. When VNF-FGs of end-to-end network services are realized, NFVI resources across multiple NFVI-PoPs and WAN resources that connect among them should be allocated to the VNF-FGs. It depends on the target KPIs of individual VNF and VL instances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

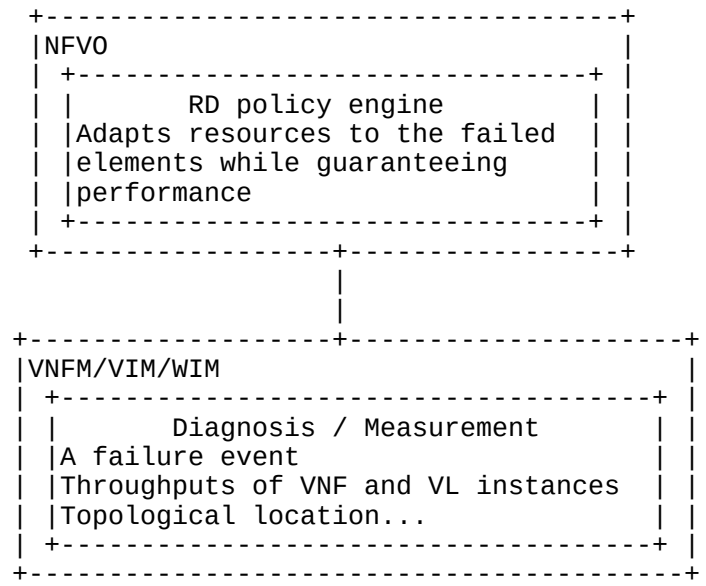


Figure 12: Architecture for policy-based fault management

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.

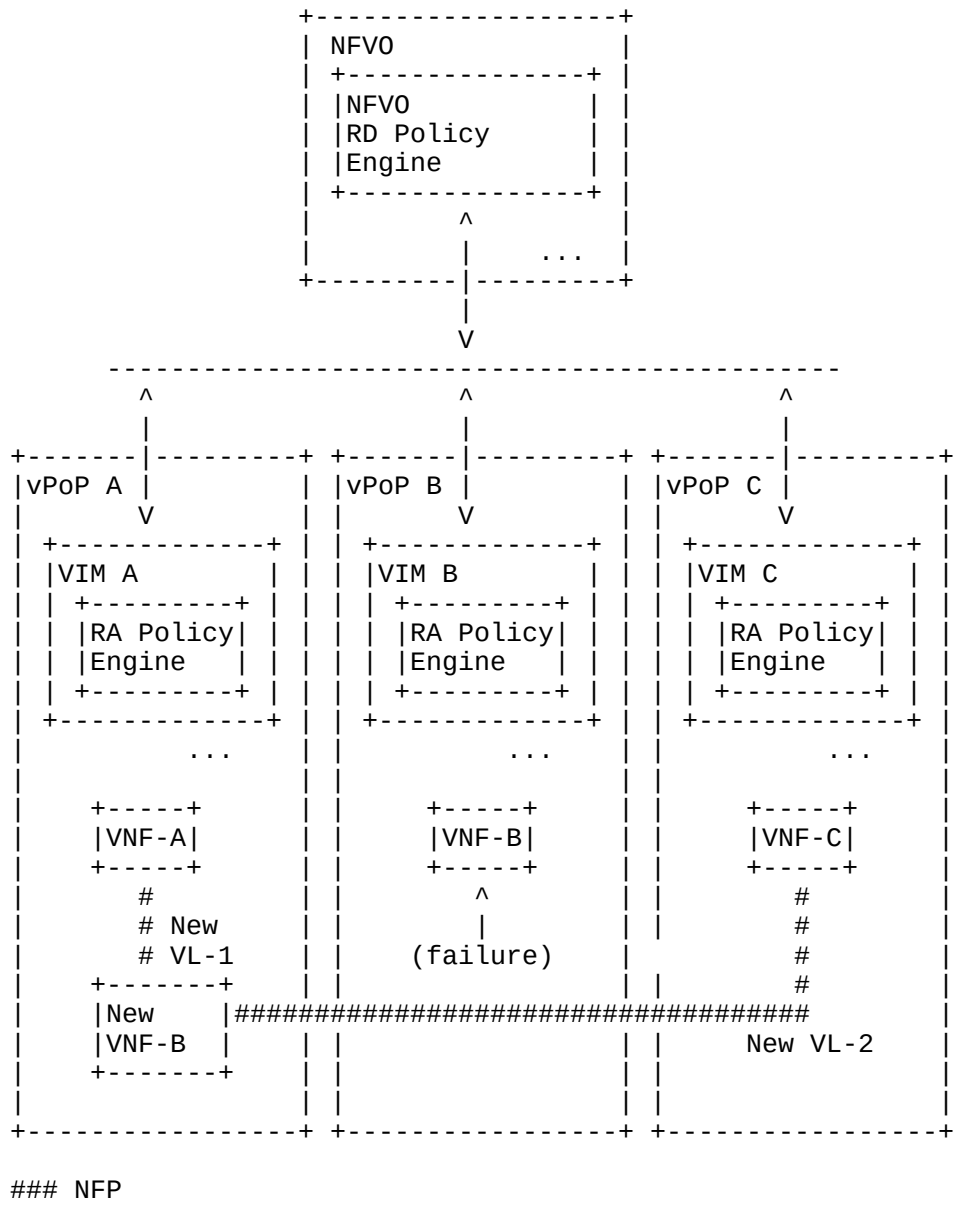


Figure 13: Re-instantiation for VNF-FG

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.

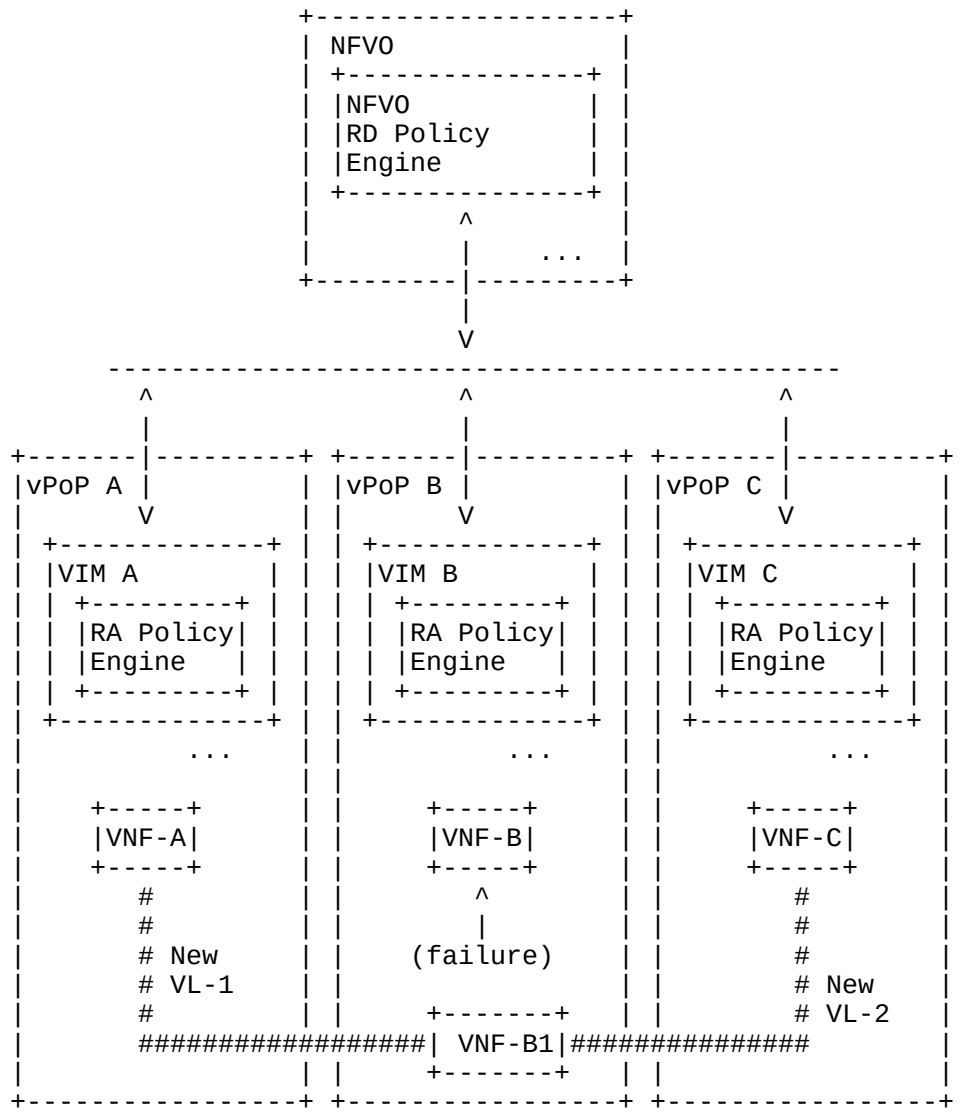


Figure 14: No Re-instantiation for VNF-FG

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_Int
ent_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: April 29, 2018

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

October 29, 2017

Verification of NFV Services : Problem Statement and Challenges
draft-irtf-nfvrg-service-verification-05

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 December 27, 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 end-to-end 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]. These 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 VNFs can be composed together to reduce the complexity of 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 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 are located 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 to Y, 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 an 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 be located 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 experiencing congestion, the end-to-end performance (e.g., latency and throughput) in the VNF FG can be degraded. Furthermore, policies on the performance such as minimum bandwidth or latency can be given to VNFs or their FGs. 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 Network Service (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 inconsistent 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 information on the entire network states and resource usage 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 able to 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 projects 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 policing feature/approach/mechanism has been added in open source projects [ODL],[ONOS]. 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 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 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. Complexity of live traffic verification

It is a 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, all the information on VNFs including configurations, dynamic states and their temporal orderings need to be precisely expressed in platform independent languages based on formal semantics. The languages and their semantic models should be optimized to the verification frameworks as well as the NFV infrastructures.

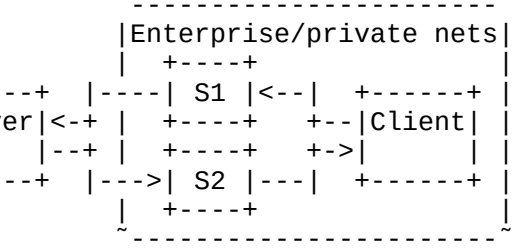
5.6. Stateful VNFs with multiple physical views

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.

- 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, the 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.



- 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. 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.

6.1.3 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.

6.1.4 VSPerf

The VSPerf projects provides an automated testing framework and comprehensive test suite based on industry standards for measuring data-plane performance. The architecture of VSPerf is agnostic to switch and traffic generator and test scenarios can be customized.

The VSPerf can be used for developing switching technologies as well as for evaluation and optimization of the data-path performance.

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 spread 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. Also thank Jose Saldana for his useful comments.

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/

[ONOS] "Open Network Operating System, "http://onosproject.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

NFVRG
Internet-Draft
Intended status: Informational
Expires: May 20, 2018

M. A. Vazquez-Castro
T. Do-Duy
UAB
S. P. Romano
A. M. Tulino
Unina

November 16, 2017

Network Coding Function Virtualization
draft-vazquez-nfvrg-netcod-function-virtualization-02

Abstract

This document describes network coding as a network function. It also describes how a network coding function can be virtualized and integrated with virtual network functions architectures. The network coding function is not a traditionally implemented network function in dedicated hardware as those that have triggered network function virtualization. It refers to a novel network functionality that generalizes classic packet-level end-to-end coding. Classic packet-level end-to-end coding helps in the provision of quality of service by trading off delay and reliability. Network coding goes beyond that by enabling in-network optimized re-encoding, which can provide both throughput gains and diverse network-controlled degrees of reliability. Consequently, a virtualized network coding function can serve as a flow engineering tool over virtualized networks (e.g. over network slices).

Table of Contents

1. Introduction

Network coding(NC) is a novel technology that can be seen as the generalization of classic point to point coding to coding for network flows. As with classic coding, both information theoretical and algebraic codes literature provide the conceptual solid basis of NC. Such conceptual basis has clarified NC benefits and corresponding tradeoffs, which need to be considered in practical implementations of the technology.

NC does not replace end-to-end (packet-level block) coding, which is a well-established technology for the per-flow provision of quality of service by trading off delay and reliability. Instead, NC provides coding within and across network flows relying on in-network re-encoding based on service-intent-oriented policy strategies. By means of such policy strategies, the provision of quality of service that NC can offer can be tailored according to desired network service intent.

For its operation, NC relies on having access to network, computation and storage resources throughout the network. Such novel networking, computational and storage ingredients of a coding technology calls for novel practical design approaches to truly exploit the potential capabilities of NC.

On the other hand, Network Function Virtualization (NFV) and NC can be seen as different ways to address different challenges in the design of upcoming network technologies. Moreover, NC is not a traditionally implemented network function in dedicated hardware, which are the network functions that have triggered the design of NFV architectures. However, in this document we show the feasibility and benefits of virtualizing the network coding function.

The objective of this document is not to explain network coding technology. The interested reader should find this information outside this document.

The objective of this document is fundamentally two fold. First, we show that NC can be designed as a (modular) network function. The modularity is convenient for the user and is given as sets of elementary functionalities (toolboxes) that are defined independent of the physical network. Second, we show that the NC function requirements of connectivity, computation and storage resources find a natural practical design solution in the integration of the NC function with available NFV architectural frameworks. Such solution is described here and it combines network protocol-driven and system modular-driven design approaches.

The resulting Virtual Network Coding Function (VNCF) can be useful for upcoming networking needs derived from network virtualization.

In order to provide the readers with a flavor of how the ideas presented in this draft might be applied to real-world communication scenarios, we will describe an interesting use case related to the creation of a hybrid satellite-terrestrial infrastructure for the effective delivery of multimedia contents to end-users. The architecture in question envisages a combination of multicast, simulcast and unicast communication scenarios where satellite links are exploited to support local in-network caching. The satellite acts as the interconnection link between distributed in-network caches and terrestrial CDN (Cotent Delivery Network) and/or feeds edge-network caches at micro-centre locations.

The example architecture will be orchestrated through an enhanced NFV management framework exposing Network Coding functionality as a Virtual Network Function (VNF). Such a function will in our case implement a novel "combined coding" technique targeting the optimization of multicast-enabled transmissions in the presence of caching. More precisely, it will leverage cutting-edge solutions for decentralized random caching which, combined with an original content distribution technique based on coded multicast, will allow us to abtain "order-optimal" performance.

In a nutshell, the above mentioned technique allows us to somehow multiplex multiple transmission chunks on a single packet, thus enabling us to meet the twofold objective of optimizing the use of the broadcast communication medium while at the same time dramatically increasing the security level of satellite-enabled transmissions, by making them resilient to network attacks like snooping and eavesdropping.

2. Conventions used in this document

The following terms defined in this document can be found in the ETSI NFV [etsi_gs_nfv_002_v1.2.1] and the IETF [I-D.irtf-nwcrg-network-coding-taxonomy].

Coherent Network Coding: Source and destination nodes know network topology and coding operations at intermediate nodes.

Noncoherent Network Coding: Source and destination nodes do not know network topology and intermediate coding operations. In this case, random network coding can be applied.

Flow: A stream of physical packets logically grouped from the network coding perspective. These packets may come from the same application

(in that case they are identified by the five-tuple: source and destination IP address, transport protocol ID, and source and destination port of the transport protocol), or come from the same source host (in which case they are identified by the 3-tuple source and destination IP address, Type of Service (TOS) or Diffserv code point(DSCP)). This distinction depends on the use-case where network coding is applied.

Intra-flow coding: Network coding over payloads belonging to the same flow.

Inter-flow coding: Network coding over payloads belonging to multiple flows.

End-to-end coding : Transport stream is coded and decoded at end-points.

Coding node: Node performing coding operations.

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.

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

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.

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

3. Network coding as a network function

3.1. Design domains of the network coding function

NC design involves different domains. There are three reasons to identify such different domains for the design of network coding functions.

First, network coding is intrinsically multidisciplinary involving at least dealing with the design of codes and networking using codes. Therefore development of solutions can benefit from a clear distinction of in which domain experts are contributing.

Second, a network coding function is a transversal network function that can be used to provide solutions to different types of problems such as congestion problems, bottleneck problems, losses problems, security problems, etc. Therefore, there should be more design domains other than purely protocol domain as it is the case with standard protocols.

Finally, a network coding function that will operate over softwarised networks with cloud storage and computational resources, needs to be designed in a way that is close to a functional software architecture.

We identify at least the three domains, as illustrated in the following subsections.

3.1.1. Coding domain

This is th domain for the design of network coding codebooks, coherent or noncoherent encoding/decoding schemes, performance benchmarks, appropriate mathematical-to-logic maps, etc. This is a domain fundamentally designed by coding theorists.

[Editor’s note] To be completed...

3.1.2. Functional domain

This is the domain for the design of the different sub-functions for network coding to achieve the desired design objectives upon abstractions of networks and systems.

This domain jointly requires to consider physical-logical abstraction, identification of network coding application to either inter-flow or intra-flow network coding, service intent and related networking for the provision of quality of service.

[Editor’s note] To be completed...

3.1.3. Protocol domain

This is the domain for the design of headers, initial settings, etc for the physical transporting of the network coded information flow as one way or interactive protocols.

[Editor’s note] To be completed...

3.2. Flexible modular design via sets of subfunctions

In order to provide the designer with sufficient flexibility, NC elementary sub-functionalities can be grouped in the functional domain as a set of toolboxes that the designer can use.

We define the three toolboxes described in the following subsections.

3.2.1. Coding/Re-encoding/Decoding Functionalities (CRDF)

[Editor’s note] To be completed...

3.2.2. Flow Engineering Functionalities (FEF)

These subfunctionalities perform optimization of available network resources to optimally perform NC to meet the service design targets depending on the (statistical) status of the networks (congestion, link failures, etc).

[Editor’s note] To be completed...

3.2.3. Physical/Abstraction Functionalities (PAF)

These subfunctionalities performing interaction with available storage and computation physical resources that are abstracted by the other toolboxes.

[Editor’s note] To be completed...

4. Virtual Network Coding Function

4.1. Virtualization of flows

An important differentiating aspect of NC with respect to traditional networking technologies is the following. A network flow for a NC network function is understood as a stream of physical packets logically grouped from the network coding perspective.

NC can optimize the NC operation abstracting such physical flow as a mathematical model, which can be subject of computational

manipulation. This makes NC to be naturally integrated into a virtualized framework of abstract entities such as virtual network or network slices. This is because in the NC case, not only the network and resources are abstracted, but also the stream of packets is abstracted.

Consequently, when interpreting NC as a functionality provided to the network, NC function virtualization simply consists of integrating the NC functional toolboxes described in the previous section into existing architectural NFV frameworks. The virtualization of the network flow is managed by the NC function (CRDF toolbox), and the virtualization of all the functionalities described in Section 3 has no difference with respect to any other network function.

4.2. Integration with ETSI NFV architecture

Figure 1 shows our proposed virtual NC network function (VNCF). It is integrated with the ETSI NFV architecture given the abstracted underlying physical system/network as part of NFVI.

The integration naturally includes too exchanges between VNCF and NFV-MANO over reference points.

Clearly, the functionalities of the FEF toolbox need to interact with the NFVO, VNFM, and VIM. Note that the NFVO two main responsibilities of orchestration of NFVI resources across VIMs and the life-cycle management of network services, fit perfectly the needs of the FEF and PAF toolboxes. Specifically, the FEF can obtain available network, connectivity and computation resources, geo-statistical status of the networks such as congestion, link failures, etc. With these, NC operation can be optimized to meet the service design targets given the service-specific design constraints. The optimization may result into manipulation of the (non-physical) flows and other flow engineering policies. On the other hand, the FEF can interact with the VIM to obtain the allocation, upgrade, release, etc of NFVI resources.

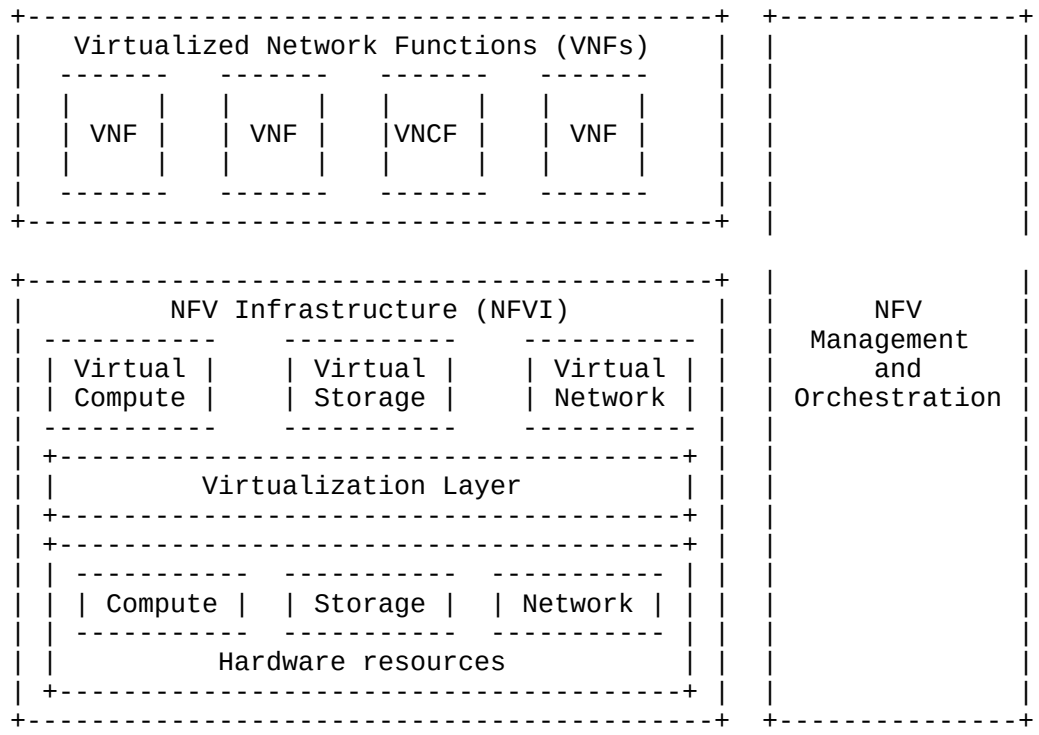


Figure 1: ETSI NFV framework with one VNCF box as part of the set of available VNFs

4.3. Example

We describe here a high-level example of a general procedure of interaction between the VNCF and the NFV-MANO. The NFV-MANO has repositories that hold different information regarding network services (NSs) and VNFs (VNCF is part of VNFs). There are four types of repositories as follows:

- o VNF catalogue represents the repository of all usable VNF packages, supporting the creation and management of the VNF packages.
- o NS catalogue represents the repository of all usable NSs.
- o NFV instances is the repository that holds details of all VNF instances and NS instances, represented by either a VNF record or a NS record, respectively, during the execution of VNF/NS life-cycle management operations.

- NFVI resources is the repository that holds information about NFVI resources utilized for the establishment of NS and VNF instances.

Assume a network abstracted as a set of N coding nodes, each with encoding/re-encoding/decoding and (possibly) multi-link connectivity. A user of the VNCF wants to provide an ultra-reliable service (e.g. mission-critical communications) to the N nodes. The performance objectives are given as a set of N reliability and delay objective performance metrics, which are geo-location dependent. We call this VNCF instantiation as a virtual geo-network coding function (VGNCF), which is activated and its management and orchestration take place.

A detailed interaction with the architectural blocks (some under definition) is as follows.

- TBD

The next section will briefly introduce a real-world application scenario associated woth the effective deivery of multimedia content in a hybrid satellite-terrestrial network.

4.3.1. The SHINE use case

SHINE stands for "Secure Hybrid In Network caching Environment". It has two main distinctive features, associated with, respectively, the broadcast-enabled satellite core and the edge distribution networks. Within the former part of the network, we rely on network coding in order to define a coded multicast technique allowing us to improve both performance and security of communications. At the edges of the distribution network, which also act as in-network caches, we instead leverage cutting-edge streaming technologies (namely, MPEG-DASH and/or WebRTC) in order to optimize content distribution towards the end users of the network.

A high-level view of the SHINE architecture is reported in Figure 2. The picture highlights the main logical components of the architecture, in terms of macro-blocks and related functionality. Namely, we identify the following elements:

1. a source encoder block, taking on the responsibility of properly encoding the original content in order to allow for the subsequent coded multicast transmission over the satellite network;

2. the core satellite-enabled communication infrastructure, looking after DVB-enabled transmission of coded multicast frames from the content provider to the edge caches, both during the cache

population phase and during the steady-state operation of the CDN;

3. two different "flavors"" of edge access networks: (i) a WebRTC-enabled access network, included in the architecture in order to demonstrate SHINE’s operation in the presence of this novel real-time communication infrastructure at the edges of the overall content delivery architecture; (ii) an MPEG-DASH enabled access network, included in the architecture in order to demonstrate SHINE’s capability of leveraging such a well-assessed web-based distribution approach.

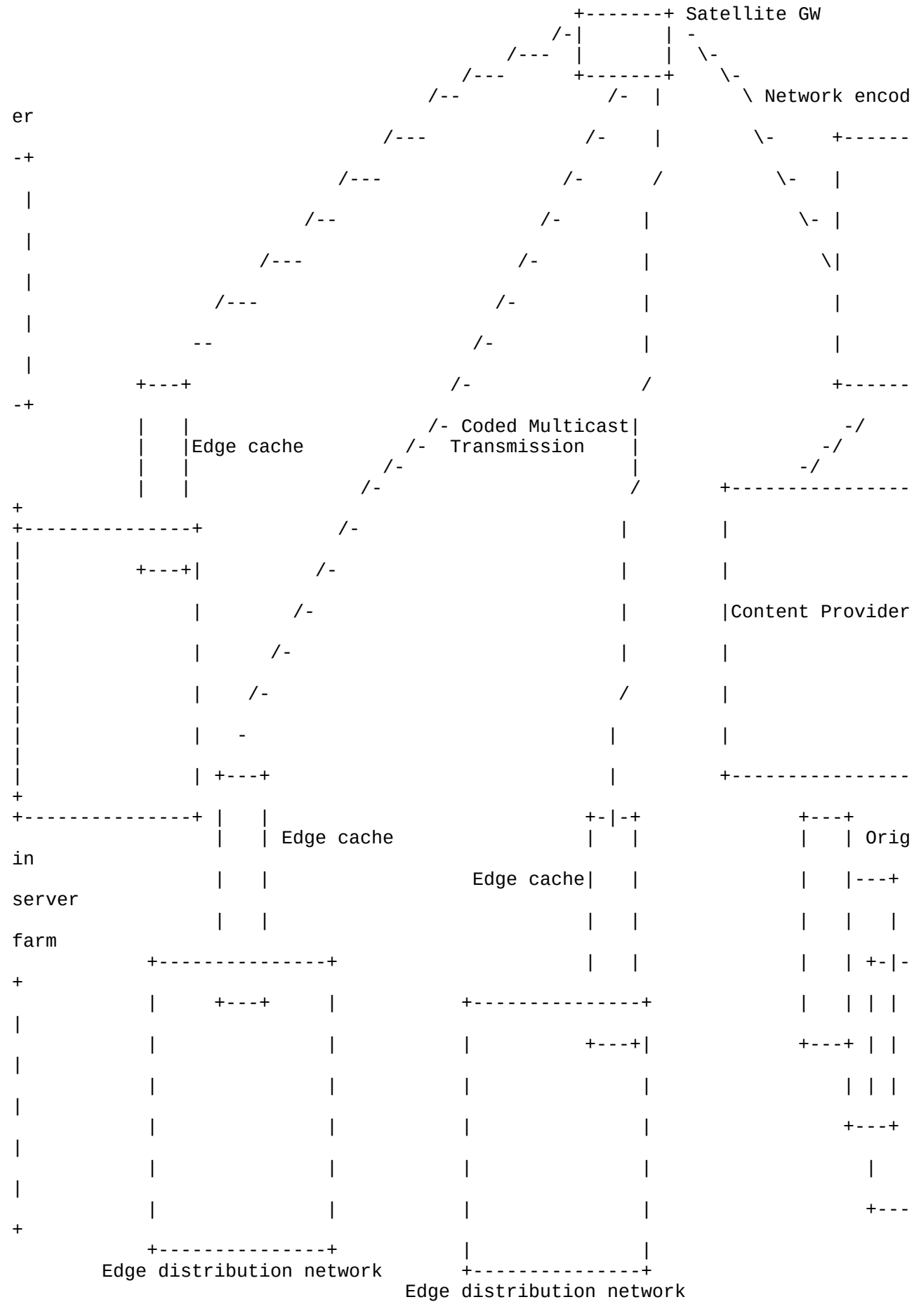

Satellite GW
Network encoder
Edge cache
Coded Multicast Transmission
Content Provider
Edge cache
Edge cache
Origin server farm
Edge distribution network
Edge distribution network

Figure 2: The SHINE use case

The system components which are of uttermost importance in this document, in view of the observation that they can highly benefit from the effective utilization of Network Coding as a Virtual Network Function are analyzed in further detail in the following.

The source encoder is a software module implementing the main logic behind the proposed coded multicast technique. It is in charge of

transforming the original content and applying the required transformations in order to arrive at a representation format that is suitable for the subsequent coded multicast transmission. The component in question has indeed to look after both the cache population phase and the actual content delivery phase. The cache population phase envisages that the edge caches pre-fetch some content, based on appropriate functions of the content library, as well as on information about estimated future users' demand for content. During the delivery phase, on the other hand, the source forms a multicast "codeword" to be transmitted over the shared link in order to meet the actual users' content demands. As already stated, we envisage that the cache population phase is carried out through transmission (over the satellite core network connecting source node with edge caches) of content chunks. As to the content delivery phase, it takes place through DVB-encapsulated transmission, over the satellite network, of coded multicast frames.

Satellite Core Network is the network segment that basically interconnects the Source Encoder, which produces and processes multimedia contents, and several Edge Networks, where the in-network caches represent the boundary network elements. The satellite network trunk leverages standard DVB-S or DVB-S2 broadcast

The delivery phase hence occurs after the placement phase, when traffic is high and network resources are scarce and expensive (e.g., in the evening). At the beginning of this phase, each user reveals its request for one of the m files. The server is informed of these K requested files. In response, the server sends RF bits (or the equivalent of R files) over the shared link. The number R is called the rate of the server transmission or equivalently load of the satellite link. From the server transmission and its local cache content, each user needs to be able to recover their requested files. As already anticipated, SHINE looks after both the content placement and delivery phases. The objective is to minimize the rate R with which every possible set of user demands can be satisfied. The constraints are the storage limit during content placement and the recovery requirement during content delivery. Both phases are generic for both coded and uncoded schemes, but naively performed in the uncoded case. In fact, when relying on uncoded or naive multicasting during the delivery phase, it is well known that the optimal caching strategy is to cache the top M most popular files at each user cache. Though, this is in general far from optimal when coding can be used in the delivery phase. Thanks to the adoption of the dynamically provided Virtual Network Coding Function, SHINE discloses the potential of caching-aided code design and illustrates its major advantages compared to the optimal caching policy under uncoded (naive) multicasting. In a nutshell, the designed architecture shows how the combined use of edge caching and coded

multicasting represents a promising approach to simultaneously serve multiple unicast demands via coded multicast transmissions, leading to order-of-magnitude bandwidth efficiency gains.

5. Conclusions

This memo presents a preliminary version of proposal for the design of NC as a network function. It is also discussed that it can be virtualized and integrated into a NFV architecture.

6. Differences with respect to version -01

Major restructuring of section 3.

7. Acknowledgements

The authors want to thank Dr. Harald Skinnemoen for useful comments and discussions. The first author wants to thank Dr. Carlos J. Bernardos and Luis M. Contreras for useful discussions.

The authors also want to acknowledge the following ongoing projects.

1. GEO-VISION - GNSS driven EO and Verifiable Image and Sensor Integration for mission-critical Operational Networks. EU funded project under the call H2020-GALILEO-2014-1 by the European Global Navigation Satellite Systems Agency (project reference 641451).

2. SatNetCode - Satellite Network-Coding for high performance, semantic-aware mission-critical visual communications. This project is funded by the European Space Agency, under contract No. 4000115046/15/NL/US.

3. HENCSAT - Highly Efficient Network Coding for Satellite Applications Test-bed. This project is funded by the European Space Agency, under contract No. 4000118143/16/NL/EM.

4. SHINE - Secure Hybrid In Network caching Environment. This project is funded by the European Space Agency, under Contract No. 4000118273/16/NL/CLP.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

This memo includes no Network Coding Function Virtualization - specific security definitions yet.

10. References

10.1. Normative Information References

[etsi_gs_nfv_002_v1.2.1]
"Network Function Virtualisation (NFV); Architectural Framework", 2014.

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

[I-D.irtf-nwcrg-network-coding-taxonomy]
Firoiu, V., Adamson, B., Roca, V., Adjih, C., Bilbao, J., Fitzek, F., Masucci, A., and M. Montpetit, "Network Coding Taxonomy", draft-irtf-nwcrg-network-coding-taxonomy-01 (work in progress), October 2016.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, 2011.

10.2. Conceptual ground basis

[AHL00] Ahlswede, R., Cai, N., Y. R. Li, S., and R. W. Yeung, "Network information flow", in IEEE Trans. Inform. Theory, vol. 46, pp. 1204-1216, July 2000.

[KOE03] Koetter, R. and M. Medard, "An algebraic approach to network coding", in IEEE/ACM Trans. on Networking, vol. 11, n. 5., pp. 782-795, October 2003.

[LI03] Y.R.Li, S., W. Yeung, R., and N. Cai, "Linear network coding", in IEEE Trans. Inform. Theory, vol. 49, n. 2., pp. 371-381, February 2003.

10.3. Application references

[ALE13] Alegre-Godoy, R. and M. A. Vazquez-Castro, "Spatial Diversity with Network Coding for ON/OFF Satellite Channels", in IEEE Communications Letters, vol. 17, No. 8, pp. 1612-1615, August 2013.

[ALE15] Alegre-Godoy, R. and M. A. Vazquez-Castro, "Network Coded Multicast over Multi-beam Satellite Systems", in Mathematical Problems in Engineering, vol. 2015, Article ID 364234, May 2015.

[DO16.1] Do-Duy, T. and M. A. Vazquez-Castro, "Design of Virtualized Network Coding Functionality foR Reliability Control of Communication Services over Satellite", submitted to Special Issue on Network Coding. International Journal of Satellite Communications and Networking, 2016.

[Do16.2] Do-Duy, T. and M. A. Vazquez-Castro, "Network coding function virtualization", in IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), September 2016, INVIED PAPER.

[HAN15] Hansen, J., E. Lucani, D., Krigslund, J., Medard, M., and F. H. P. Fitzek, "Network coded software defined networking: enabling 5G transmission and storage networks", in IEEE Communications Magazine, 2015.

[HEI09] Heide, J., V. Pedersen, M., H. P. Fitzek, F., and T. Larsen, "Network Coding for Mobile Devices - Systematic Binary Random Rateless Codes", in ICC Workshops, 2009.

[SAX15] Saxena, P. and M. A. Vazquez-Castro, "DARE: DoF-Aided Random Encoding for Network Coding over Lossy Line Networks", in IEEE Communications Letters, vol. 19, No. 8, pp. 1374-1377, August 2015.

[SZA15] Szabo, D., Nemeth, F., Sonkoly, B., Gulyas, A., and F. H. P. Fitzek, "Towards the 5G revolution: A software defined network architecture exploiting network coding as a service", in SIGCOMM Comput. Commun, 2015.

[VAZ15.1] A. Vazquez-Castro, M., "A Geometric Approach to Dynamic Network Coding", in Information Theory Workshop, Jeju, Korea, October 2015.

[VAZ15.2] A. Vazquez-Castro, M., "Subspace coding over Fq-linear erasure satellite channels", in 12th International Symposium on Wireless Communication Systems, pp. 216-220, August 2015.

[VAZ15.3] A. Vazquez-Castro, M. and P. Saxena, "Network Coding over Satellite: From Theory to Design and Performance", in Volume 154 of the series Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 315-327, September 2015, INVITED PAPER.

Authors' Addresses

M.A. Vazquez-Castro
Autonomus University of Barcelona
Campus de Bellaterra
Barcelona, 08391
Spain

Email: angeles.vazquez@uab.es

Tan Do-Duy
Autonomus University of Barcelona
Campus de Bellaterra
Barcelona, 08391
Spain

Email: tan.doduy@uab.es

Simon Pietro Romano
University of Napoli Federico II
Via Claudio 21
Napoli, 80125
Italy

Email: spromano@unina.it

Antonia Maria Tulino
University of Napoli Federico II
Via Claudio 21
Napoli, 80125
Italy

Email: antoniamaria.tulino@unina.it

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: June 3, 2017

A. Veitch, Ed.
Netcracker, Inc.

November 30, 2016

Use Cases and Analysis on Integrated NFV and Network Optimization draft-veitch-nfvrg-nfv-nw-optimization-00

Abstract

This is a review of issues related to the optimized deployment of network services, aka service function chains, composed of virtual and physical network functions, where these functions may be instantiated in multiple distributed data centers. Criteria for optimization are introduced, and use cases are described and expanded, in order to establish the need for coordinated computation of NF deployment sites and the connectivity among them. Some methods for addressing optimization pain points are described, and potential new requirements of the PCE and PCEP are discussed.

One point to make clear is that the exploration of specific algorithms for NF/VNF placement and path computation is outside the scope of this draft. We are not looking at computational or optimal greedy search algorithms. The goals for this draft are 1) to provide justification through use cases for more tightly integrating path and placement computation algorithms, and 2) looking at how some of the optimization requirements might be addressed, mostly through pre-computation and caching, and how these might affect management and orchestration functions and the protocols (e.g. PCEP) that are used.

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 June 3, 2017.

Table of Contents

1. Introduction

With the advent of Network Function Virtualization (NFV), network services, consisting of an ordered set of network functions (NFs), may be composed of physical (PNF) and virtual network functions (VNF). These VNF will execute in one or more virtual machines or containers operating on standard high performance servers. The set of network functions for a given network service might be instantiated within a single site or the network service might be multi-site.

This document is a brief examination of issues related to the optimized deployment and operations of network services, and in particular, multi-site network services. It includes a review of use cases that reveal possible limitations in current NFV and SDN management architectures, as defined in some standards and open source projects, and related communications protocols.

The use cases described include the virtualization of support for Content Delivery Networks (CDN), Internet of Things, video narrowcasting, and high performance computing. In this analysis, an important consideration is the shared infrastructure among a variety of applications and services. Sharing the infrastructure has the advantage of increasing overall utilization of the infrastructure vs. support for dedicated solutions for each service type. One can imagine disjoint networks for each of the services being implemented, with the result being a very high total cost to the provider and consumer.

Another advantage is that the resource utilization profiles for each of these services will differ. High performance computing might require a good deal of computational power and memory, which translates into requiring many CPU, virtualized CDN require a large amount of storage, and the main requirements for video services might be low latency and jitter in the delivery of its traffic from its source to its consumers.

An understanding of the expected resource usage for different services and network functions will allow intelligent placement of these services, not just to provide the user a high quality of experience (QoE), but also provide the operator with a lower CAPEX through smarter utilization of the resources.

One of the goals in reviewing these use cases is to understand the common information that is necessary for an operator to effectively provide optimal solutions for the various services to be supported, and their disparate requirements.

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. Definitions

Note: This document includes concepts and terminology used in both the IETF and in the ETSI NFV ISG. A number of concepts are shared or have very similar counterparts between the groups. Here are some of the key terms used and their definitions.

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

- Network Function (NF) - A functional block within a network infrastructure, that has well-defined external interfaces and a well-defined functional behavior.

- Virtual Network Function (VNF) - Implementation of an NF that can be deployed on a Network Function Virtualisation Infrastructure (NFVI). IETF corollary is the Service Function or SF in SFC.

- NFV Infrastructure (NFVI) - The NFV-Infrastructure is the totality of all hardware and software components which build up the environment in which VNFs are deployed. It may span across several sites, e.g. where data centers operate.

- NFVI-PoP - Where a Network Function is or could be deployed as Virtual Network Function (VNF)e

- Network Function Virtualization Orchestrator (NFVO) - functional block that manages the Network Service (NS)lifecycle and coordinates the management of NS lifecycle, VNF life cycle and

NFVI resources to ensure an optimized allocation of the necessary resources and connectivity

- Network service (NS) - A composition of network functions and defined by its functional and behavioral specification.

- VNF Forwarding Graph (VNFFG) - A NF forwarding graph where at least one node is a VNF.

- Multi-site network service - A service that has component network functions operating in infrastructure located in separate geographical sites. These sites may consist of a combination of customer sites and operator data centers. Service components are therefore connected across wide area networks (WANs.)

4. Optimization Criteria

One question to be asked, when we mention optimization, is, what is it that we want or need to optimize when a network service or services are to be deployed? And the answer is that there are a number of criteria we would like to see maximized or minimized, some of which lead in opposite directions.

- Maximize the likelihood a requested network service will be honored and instantiated, and not rejected due to a lack of available resources to meet the service's SLAs.

  For a multi-site service, it is important that the site of the network functions and the connectivity between them be selected to ensure any SLA requirements are met. The search for a solution should not fail when adequate resources exist to support the service request.

- Maximize the health of a network service and minimize the likelihood that service might fail to meet its service level agreements, or SLAs, during its lifetime.

  Each network service deployed must meet an SLA, with a penalty likely imposed should the service fail to do so. These SLAs will include metrics such as Availability, throughput or bandwidth, loss, latency, maximum outage time, mean time between failures, etc. Goals include minimizing the risk of any single service failing to meet its SLAs, as well as minimizing the overall number of services that fail to meet their SLAs.

- Minimize the time it takes to respond to a network service request with an instantiation of that service. Alternatively, a request may ask only for the set of resources to support a network

service, but not to actually deploy the services. The response time for this request should also be minimized.

- Maximize the utilization of the network and compute infrastructure, or NFVI, to achieve OPEX and CAPEX savings.

  One promise of NFV is the reduction of OPEX and CAPEX. In other words, with greater flexibility and finer grained control, the better a management system can utilize the NFVI. CAPEX is clearly lower when less total NFVI is used, and OPEX should be too. For example, excess hardware may be hibernated when not needed, saving energy costs and management costs.

5. Challenge of Network Service (or Service Function Chain) Optimization Across Multiple Sites

When deploying a network service, it is important to place the component network functions (PNF/VNF) in sites to deliver an optimized network service, as based on the criteria listed above. Deciding how and where to deploy the VNFs is made more complex when the network service is to be deployed across multiple sites.

A network service descriptor includes a set of NFs and links that represent connections between pairs of the network functions. Associated with each link might be a set of criteria, e.g. latency, bandwidth, etc., that an instance of the service must meet. When NFs are to be placed in different sites and separated by one or more networks, both the sites of the NFs and the connectivity paths between the pairs must be selected to meet the NF and connectivity requirements defined for the network service and its components. This means that choosing the placement of the VNFs and selecting their connectivity are interdependent activities.

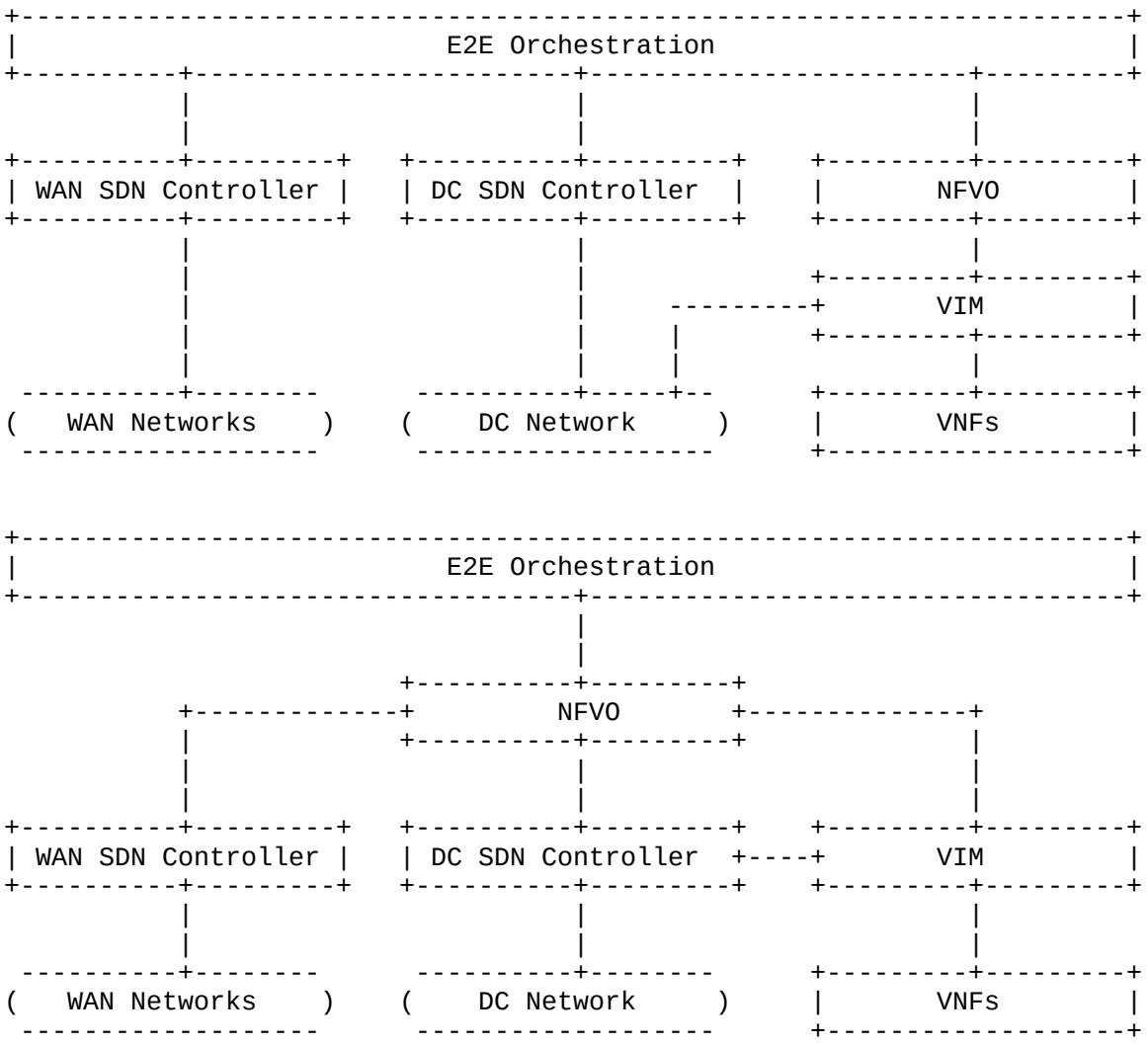
6. Review of Current NFV / SDN Management Solutions

Today a number of management and orchestration architectures for NFV based services have been defined and are described by standards organizations, open source projects, as well as some service providers. The ETSI NFV ISG described the original NFV management architecture: The Management and Orchestration of NFV based network services, commonly referred to as MANO, was defined in phase 1 of the ISG work program.

Since then, other standards bodies have addressed the architecture, and a number of software projects undertaken to develop management systems, based on, or strongly influenced, by the ETSI MANO architecture. These include, but are not limited to:

- Lifecycle Service Orchestration (LSO) from the Metro Ethernet Forum (MEF)
- Open Source MANO (OSM), also sponsored by ETSI
- Open Platform NFV (OPNFV) from Linux Foundation
- SONATA
- Openstack, including Tacker, and networking-sfc
- Open-O
- Enhanced Control, Orchestration, Management and Policy (ECOMP) from ATT
- TMForum architecture

These architectures generally follow two paradigms. Each includes an End-2-end (E2E) orchestrator that is responsible for overall orchestration of network services, including the VNFs, PNFs and network connectivity. This E2E orchestrator then communicates with an NFVO, an orchestrator for the network services based on the VNFs. Where they differ is in how the network controller, often an SDN controller, fits. In most of these, the E2E controller communicates directly to both the NFVO and the network controller. In others the NFVO communicates to the network controller.



Two typical paradigms of network service orchestration seen in open source and SDO architectures.

Figure 1

The key point is that in both paradigms the NF deployment decisions and connectivity decisions are controlled by separate functions. As described in literature to date, the functionality described for each, and the information to be exchanged, is not adequate to avoid potential inefficiencies and unnecessary network service deployment

failures, due to the real interdependence these functions have upon each other.

Of course, these architectures are currently undergoing active development through open and iterative design processes. Therefore, the state of these designs is very fluid, with new capabilities continuously evolving. Therefore, the description provided here represents only a moment in time. This document is intended to provide insight into possible limitations that exist in computing deployment solutions and to recommend updates to functional block capabilities and communications (perhaps leading to protocol updates), that will lead to improving these computations. It is expected that as these projects evolve, this draft too will evolve, further clarifying roles, communications, and needed extensions.

7. Use Cases

7.1. Introduction

The following are a set of network service examples that demonstrate the value of coordinated or unified compute and network deployment and configuration planning.

7.2. Virtual Content Delivery Network (vCDN) - Dynamic (Flash) Delivery

The following are a set of network service examples that demonstrate the value of coordinated or unified compute and network deployment and configuration planning.

ETSI and other groups have identified virtual CDN (vCDN) as an application well suited to operate as an NFV network service. As 5G is adopted and bandwidth greedy applications such as UHD video proliferate, mobile data will have a huge impact as a source of network traffic. CDNs have a proven track record as an effective way to provide high quality and low latency delivery of content to users, while simultaneously limiting the overall utilization of network bandwidth. Operators need to continue to provide a high quality service while limiting provisioning unnecessary infrastructure. Virtual CDNs are seen as an excellent way to address these new pressures.

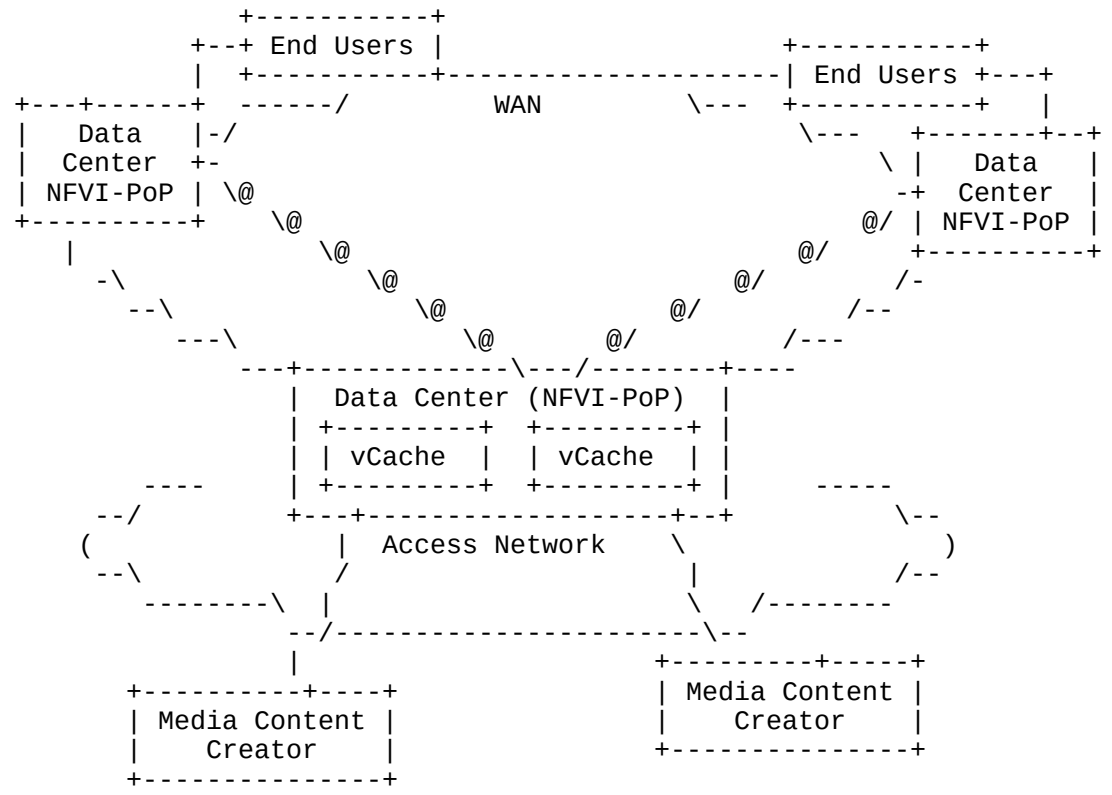
There are some obvious advantages to a virtualized solution. The costs will shift from CAPEX to OPEX as the vCDN can operate on standard high capacity servers. More importantly, a software-based solution enables rapid configuration of the vCDN, faster upgrades of the software, and mobility of the vCDN vCaches. With vCDN it is possible to dynamically and intelligently size the vCaches, and move them to sites where they provide the best value, based upon the

current and anticipated near-term traffic volumes and patterns. More ephemeral demands for content can be more effectively managed.

The SONATA Use Cases and Requirements document (http://www.sonata-nfv.eu/content/d21-use-cases-and-requirements) categorizes two types of vCDN scenarios.

First is the more traditional distribution of popular content. Content originates at a content provider and it is distributed on vCaches located as needed for a large number of subscribers.

The second scenario is concerned with user-generated content and is much more dynamic. This is the one that is examined here. This is a 'flash' vCDN, one where the need for a CDN is due to a sudden burst of content, one with a limited expected lifetime. One example might be any large sporting event that many people attend. These attendees will likely take and post multiple videos to the Web with a social media application, and then they and their friends will view and download these videos multiple times. During the event there will a real need to support caching this content and making it available to consumers while limiting network impact.



Topology of vCDN use case with a rapid short term need for content caching.

Figure 2

The challenge for this use is for the provider to be able to rapidly deploy and configure the a vCDN solution so that the users' QoE is high, both the sources and the consumers, while the resource utilization, mostly the WAN connectivity, is minimized.

In this use case the placement of the vCaches is driven by a number of factors, including the proximity to the users posting the videos, the proximity to users consuming the videos, the available network capacity among the data centers, reliability (availability) depending upon the service, the costs of the data center servers, and the cost of the network, if connectivity requires transit across a third party's network. This is not simply a matter of identifying the data centers and then connecting them. Candidate sites must be identified, connectivity options must be identified, and the combination of these must be evaluated in the context of the factors

listed above, past and current performance measurements, and any provider or user policies. Multiple deployment options may be considered, with one selected to provide an optimal solution.

As this will be a dynamic vCDN deployment, frequent monitoring of demand and delivery metrics is a major necessity, and redeployments and upgrades, or possible reductions when demand wanes, must be addressed with no unnecessary delay.

7.3. Video Narrowcasting Use Case

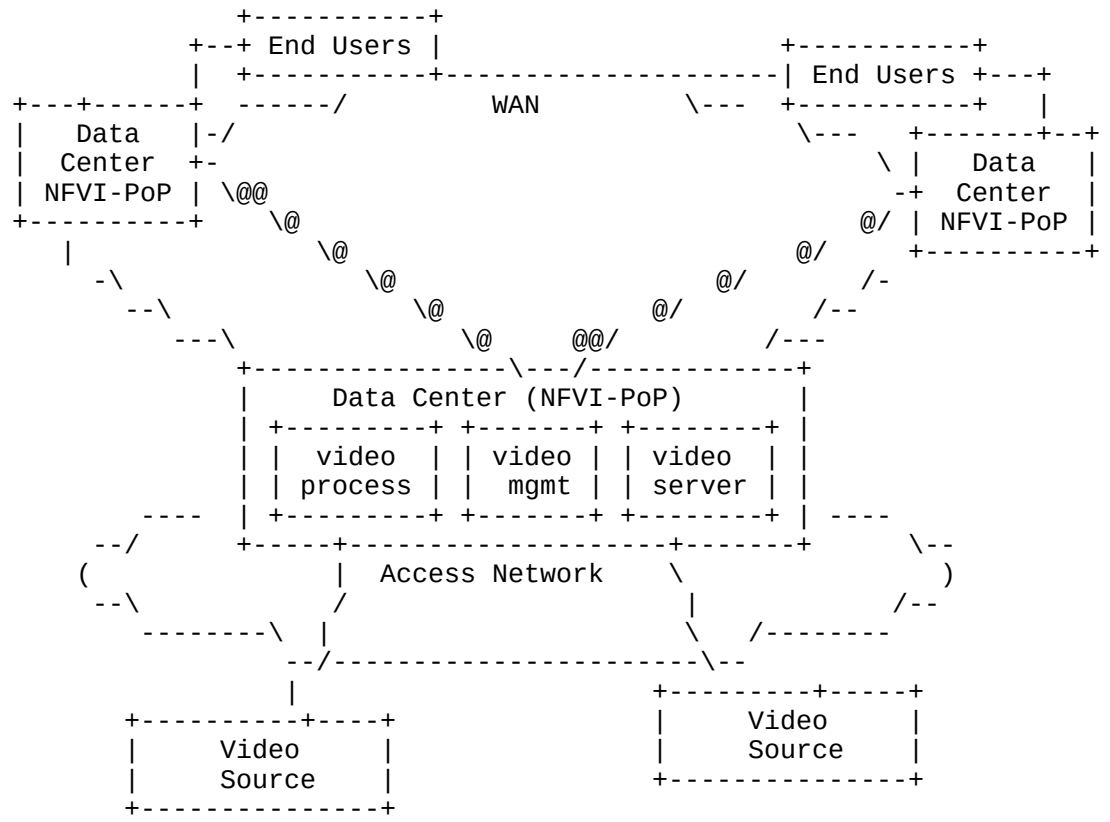
Video narrowcasting is a targeted delivery of video (and audio) content to multiple consumers, perhaps from the tens to low thousands, and delivery is generally expected to be broadcast quality. This is not broadcast television, but might be used for distance training, or education, or internal communications within large organizations. It is also a term used to describe the delivery of commercial television to a niche audience, e.g. from a cable service provider. This differs from popular microcomputer based applications such as Skype or weChat, as high quality (high definition), high availability and very low latency are expected.

In some cases, there will be more than one video source; you can imagine a presentation where presenters are based in different sites, and in some of these cases, the video encodings might differ from the different sources. In addition, lower quality video and audio might be available for users to send feedback to the primary sources.

A number of factors must be examined when choosing how to support this service. These include, the number of sources, their sites, their encoding capabilities, and the bandwidth available over which each can send. Also to be considered are the receivers, their number and site, their bandwidth capacity over which to receive the videos, and their codecs. How many incoming video streams can the receiver process at once.

Today there are commercial hardware and software products and packages to provide this capability. These, however, can be costly and are not very flexible.

You can imagine a number of situations. The simplest would be if all senders have the same video codec, all receiver codecs can decode as many incoming video streams as needed, and all senders and receivers have more than enough bandwidth for sending or receiving, and high usage of WAN links is not a problem. When this is true there is very little service management needed. Unfortunately, this is unlikely.



Topology of video narrowcasting>.

Figure 3

More likely is that some of the following are true and must be addressed.

There are multiple senders with different coding capabilities, or different bandwidths available over which to send. In addition, some of the receivers do not have the video codecs to decode some of the sender encodings, or lack the bandwidth over which to receive the packets of the high definition video.

When this is true, functions must be installed into the network to manage these issues properly. Video processing can be added to mix video of different sources, if necessary for some consumers to see multiple senders. Consumers may even remotely control this processing, on demand. Other processing may be needed to compress the video, with the effect of lowering its quality, but making it

possible for consumers with limited bandwidth (e.g. mobile) to see it. These and many other factors will drive selecting where to place these functions, connect them to one another, and how to forward these to the end consumers (e.g. what IP multicast groups, etc.), all to provide the best QoE while limiting resource consumption.

The evaluation of these factors may quickly become complex, and may be continuous as consumers join and leave. The placement of the VNFs will depend greatly on network connectivity, capacity and latency, as well as the potential to provide connections directly over electrical or optical, vs. multi-hop packet.

7.4. Real-Time Telemetry Use Case - Internet of Things

The Internet of Things (IoT) is growing exponentially and will be a huge source of information to be communicated across the Intenet in the coming years. This use case in concerned with a large number of sensors being activated over a short amount of time. One example of this could include a large sporting event, e.g. the Boston Marathon, with multiple thousands of competitors wearing health-monitoring applications that communicate vital signs as telemetry over the Internet. Or this could be relevant for a natural disaster (or man-made disaster.) Sensors might be deployed, or simply activated to report much more data, because of an incoming hurricane, or a volcanic eruption, tornado, forest fire, or tsunami and infrastructure emergencies. This could necessitate the creation of a new IoT gateway and other associated components located nearer to the sensors and with the capacity to forward the data in real-time to its consumers.

This use case is quite similar to the flash vCDN use case described above. Here, instead of vCaches, we have IoT gateways. Instead of cameras taking videos to be uploaded and downloaded, we have sensors, to be processes and forwarded in real-time to the consumers. This is a case where low latency, and perhaps low jitter, is required, though high throughput less so. Again, these network requirements might impact the placement of the gateways and their connectivity.

A further complexity is introduced when we consider these sensors to be mobile. One example will be with self-driving cars, or even human driven cars, but where there are many more sensors. The vehicles will generate a great deal of data which must travel over the wireless access networks to the nodeBs, behind which will be IoT gateways to process the data and forward the relevant data in real time to other functions. These might automatically change driving routes, maximum speeds for some roads, etc.

In this scenario it seems likely that different areas will have different loads of network sensor data based upon the time of day. There will be low volumes during the day, with greater volumes during rush hour. And these will migrate from city or industrial areas, to suburbs. (Just as, no doubt, today the cell phone calls managed in different areas depend upon the time of day.) Virtual IoT gateway deployments and the connectivity among the gateways and telemetry consumers will there for also be varied. This is more predictable, but deployment options must still consider connectivity, network utilization, etc. when placing the vIoT gateways, etc. There are also likely more dynamic mobile scenarios to be reviewed, e.g. travelers evacuating an area due to a hurricane, etc.

7.5. High Performance Computing (HPC)

Note: A number of concepts here are taken from 'HPC-Aware VM Placement in Infrastructure Clouds', by Gupta et al.

High performance computing (HPC) in the cloud promises to expand the number of computing applications that can be supported and will lower costs over all. However, HPC has a number of requirements that place significant demands upon the compute and networking infrastructure.

Included among these are the need to 1) periodically move huge volumes of data among distributed applications as well as 2) support multiple processes that are tightly coupled and require frequent communications and synchronizations. In addition, for some large computations, there may be the need to rapidly expand to very large numbers of collocated VNFs for large computations with very strict synchronization time windows.

There is active research today for improving VM scheduling, hardware utilization, how best to partition clusters for different application types, and map functions to sites to maximize utilization.

When considering some of these demands, a few things are clear. For communicating large volumes of data, it will be necessary to dedicate significant bandwidth at times and limit latency. When supporting HPC as a service a provider must anticipate these sorts of needs, and ensure the infrastructure resources supporting other services are utilized such that the capacity to meet this sort of demand is possible without the need to migrate services and VNFs.

When the resource requirements of cooperating applications recommend locating the applications in separate data centers and the application's synchronized communications requirement demands connectivity of a high bandwidth circuit, the provider must ensure both the data center infrastructure and connectivity between the

datacenters meet the service requirements. In this case, while a circuit may not exist, the function computing the connectivity should be aware of the potential circuit, for situations such as supporting this network service.

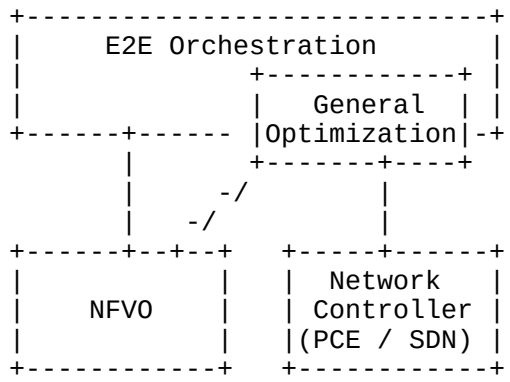
8. Orchestration of Mixed Applications and Network Services

The use cases listed above place a variety of disparate demands upon the data center (NFV-IPoP) and network infrastructure. For each individual network service instance, a service deployment algorithm may compute the sites and connectivity for that specific instance. These computations are made in the isolation of the context of that specific service instance, perhaps over its lifecycle.

However, there is a need to consider and manage overall delivery and support of these individual services. As has been described above, it is beneficial to deploy across the data centers a balanced distribution of network service components with different resource requirement profiles. This will help to optimize the utilization of the data center and network infrastructure. And in circumstances when some of the services are known to place specific large demands on infrastructure, for example, instantiating an ODU circuit to provide high bandwidth real-time communications, some agent must be aware of this sort of demand to ensure it is met.

This implies that there could be some sort of higher-level agent, one that works with the agents (e.g. NFVO and SDN controller) computing compute and network deployments for individual requested service definitions. This higher layer agent must ensure that resources are allocated properly so that those agents compute deployments that also honor the greater goals, e.g. the balanced distribution of heterogeneous services, resources ready for specific demands of some of the services, general reduced energy consumption and lower costs, etc. These should lead to the optimization goals described at the beginning of this paper.

The exact communications that might be required to support this should be explored. These could impact the controller NBI or the PCE pro tocol.



Shows an agent that tracks usage to provide guidance for general optimization.

Figure 4

9. Generalized Architecture Options for Coordinated or Unified SDN/NFV Deployment Planning

To achieve the integrated NFV/SDN optimization goals requires a software system that computes the optimizations for compute and network.

One option would be to replace the separated agents, the NFVO and SDN controller, with a single unified agent that processes the network topology, configuration and utilization, as the PCE / SDN controller does today, and computes the VNF placement, as the NFVO does today. If we look at the E2E Orchestrator in Figure 1, one option is for the network SDN controller and the NFVO to send full topology and NFVI (data center topology, compute et al resources and existing use and utilization) information to the E2E Orchestrator, where it is the unified agent, computing the optimal deployment, computing the path and NFVI allocations, and then directing specific actions of the PCE / SDN agent and NFVO to deploy the updates.

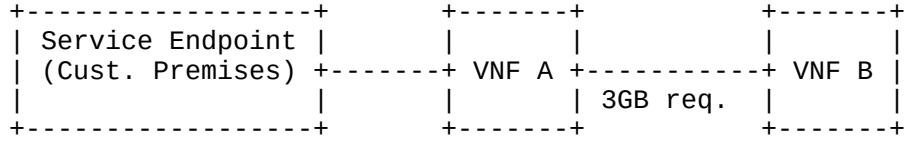
Alternatively, the functionality of the NFVO and SDN Controller remain mostly consistent. Each fills its current set of responsibilities, the PCE computing paths and the NFVO planning NFVI-PoP updates with the new VNF instances. If this is done, then enhancements must be made so these agents coordinate and compute the best overall deployment.

10. Analysis of Integrated Network and Compute Optimization with a Simple Use Case

10.1. Introduction

Here is presented a simple network service, composed of two VNFs (SFs) across two data centers. This could be a vCPE service with a remote VNF. This simple case is used here to provide a basis on which to explore how network services (SFCs) may be orchestrated. Solutions for the simple case are examined iteratively, each new version presenting more advanced orchestration capabilities that address deficiencies identified in the previous version. The purpose here is not to examine any path computation optimization algorithms. It is intended to demonstrate how other methods, including pre-computation and/or memoization, might aid in meeting the optimization goals, and what this might mean for network topology modeling and protocol definitions.

Note: Many concepts in the TE Topology model currently being defined in the TEAS WG, can be considered relevant to this work. The intention of this section is to describe the general problem and approach. How this relates to the concepts and models defined in the TE Topology is explored in the next section of this document.



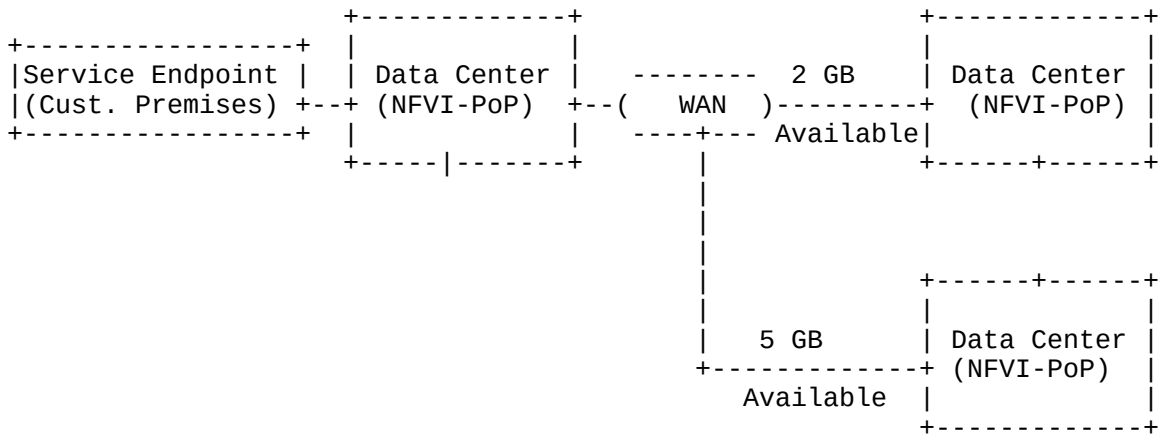
Shows an architecture.

Figure 5

10.2. Network Topology

The topology includes three data centers and a user enterprise site, where the a user service endpoint is located.

Two variations are used. In the first, tunnels are already established among the data centers. In the second, a tunnel is established specifically to support the link between the network functions.



The network topology available for establishing connectivity for the network service.

Figure 6

10.3. Simple Orchestration

In this example the computation of the VNF deployment sites occurs first, followed by the connectivity solution. There is only one link property to be met, minimum bandwidth, and for the link between VNF A and VNF B, the minimum acceptable bandwidth of 2 GB . (Note: This is shown in the labels of the links in Figure 6.)

Here are the actions and events involved in planning the creation of a service instance.

- Orchestration software receives the network service request.

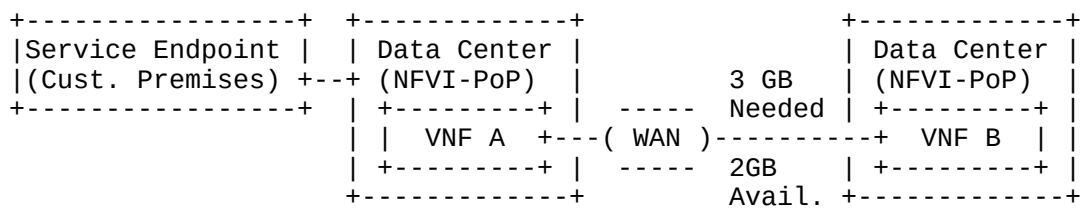
  The request includes descriptors, including resource requirements, for each VNF (SF) and the links between them. For the VNFs, this includes the number of needed containers (a VNF may be composed of multiple VNFCs), compute, memory, and storage. For the links this includes bandwidth, latency, jitter, and loss.

- Orchestrator identifies the data centers based on proximity to the user site, and the availability of servers with the resources to meet the VNF (SF) resource requirements and any relevant policies.

  At this point, the choice of data centers for the VNF has been finalized.

o Orchestrator requests the network controller to establish connectivity for each virtual link connecting VNFs in the network service description.

  A path must be found between each pair of VNF, one which can provide the minimum bandwidth associated with the virtual link between those VNF. NOTE: If no suitable path is found, then the network service deployment FAILS.



>Inadequate bandwidth for service with VNF placement computed in advance.

Figure 7

10.4. Connectivity Information in VNF Placement Planning

An obvious deficiency of this process is in choosing VNF sites without considering the ability of the network to provide the necessary connectivity and bandwidth. The solution is to update the orchestrator to consider connectivity as a factor when computing where the VNF should be placed. Whatever the placement algorithm, it must ensure that VNFs, connected by a virtual link in a network service definition, have supporting network connectivity that will meet the service requirements, e.g. bandwidth, or latency, associated with that virtual link.

Compared to the previous solution, here the network service deployment algorithm will place a VNF in a datacenter only if network connectivity is acceptable. If an orchestrator chooses to place a VNF in a datacenter, then the next VNF in the service path must be placed in a site where connectivity between the two meets the requirements in the virtual link description.

A simple backtracking algorithm, such as depth first search (DFS), will start at the front of the service and check on possible next hop data centers (including the ’current’ one). It will choose one based on proximity, available compute, storage, etc. resources, policies, etc. It will then request of the PCE or SDN controller to compute a path to the candidate data center, passing the virtual link requirements, e.g. bandwidth, latency etc. The PCE/controller should

consider the topology, currently allocated bandwidth, traffic volumes and profile, hop count, encapsulations, etc. in computing paths. This will continue until either all options have been searched unsuccessfully, or until a successful set of data centers and paths between them is found. Of course, DFS will not be used, as there are much more efficient heuristic algorithms available. But the concept is the same regardless of the algortihm. Connectivity information is utilized when choosing VNF sites.

One problem with this solution is the need to have the PCE or SDN controller compute a path each time connectivity between two data centers is to be checked. This will take a fair amount of time and CPU cycles. Efficiency can be improved through memoization of the path computation and its result, the first time the PCE or SDN controller returns a value. Before asking the PCE to compute a path, a check of the cache, for the result of any earlier computation, can be made. Keys would be the endpoint data centers, the network service ID and the virtual link ID.

10.5. Pre-computation of a Connectivity Matrix

One problem identified in the previous solution is the number of delays that accumulate due to repeated path requests to the PCE/ controller. Memoization is identified as a method for reducing the impact. Nevertheless, the response time to instantiate a service will be longer than it need be. For a network service with m VNF (SF), if each is in a separate site, then, at best, m-1 requests to the PCE/controller are required. If there are problems finding paths, the count of path computations rises. This has the potential to be computationally costly and require a lot of time before providing a satisfactory solution.

A better solution would be to pre-compute connectivity among the sites and make this available for the algorithm. This will likely increase the overall compute load, but it would shorten the time to respond to service requests. In this case a connection is computed for each pair of data centers and possible endpoints (user sites). It is maintained through continuous requests of the PCE, both triggered by a baseline frequency as well as reported events that could impact connection. Assuming a total of n data centers and other sites, this results in a connectivity matrix that is n x n. It may be sparse, as some sites may not connect to others, and of course, a site does not connect to itself. Each entry in the matrix represents a path between the sites, and each has relevant bandwidth, latency, etc. attributes.

Here now the computation, whatever the algorithm, is much more efficient. Connectivity information for any pair of data centers or

possible endpoints is locally available. There is no need to request information from another process, the PCE or other SDN controller.

10.6. Multi-valued Connectivity Matrix

Unfortunately, as described above, this solution still has limitations. Only one path, perhaps selected based on greatest available bandwidth, is computed for each entry in the n x n connectivity matrix. This path is not necessarily engineered to maximize or weight the path metrics in any way. If a path between two datacenters needs a latency of 40 msec and the path in the connectivity matrix has latency of 80 msec, that option will be rejected. This is a problem because a perfectly acceptable path with the needed bandwidth might exist along a different route.

A possible solution here is to compute more than one path for each pair of sites, where each is based on a different set of criteria priorities. These could include maximum bandwidth, minimum latency, fewest encapsulations (tunnels, tunnels in tunnels, packet layer to optical and back), fewest hops, fewest committed tunnels, least dynamic traffic patterns, etc.) Others might include different balances of these values, and anti-affinity rules, e.g. do not share the same underlying optical paths, something important for identifying connectivity with backup paths.

The result, therefore, is an n x n x m array, or simply an n x n array with each cell containing a list of path options, where the list length may vary for different sets of endpoints. Each list entry (or cell in the m dimension for a 3 dimensional array) will represent a unique path supporting the connectivity between the two endpoints, where each has been computed for a different set of service priorities.

There is still the concern that a network service might be requested for which a needed path’s requirements (from a virtual link) are not satisfied by any entry in the connectivity matrix. In other words, no connection with the relevant service requirements has been pre-computed. The implication is that this could result in an unnecessary rejection of the requested network service. To address this, it might be appropriate to include an option for the path or the orchestrator. If the deplyment algorithm finds no satisfactory path in the matrix, the orchestrator could request directly to the PCE to see if it can find connectivity to meet its needs.

Note: In the matrix, there may be endpoint pairs for which no path has been found. The value in the matrix would be FAIL, or perhaps include something with a bit more useful information. If a request

fails because the requirements do not match any of the cached paths, it is an option to ask the PCE to re-check, or not.

There are a number of ways the orchestrator could approach doing this. The orchestrator might request the PCE to compute a point to point path that meets the service requirements, while itself, IN PARALLEL, continues to compute a different VNF placement and path solution. If the PCE/controller finds a possible path to support the connection, or perhaps even if not, it would also make sense to add it to the connectivity matrix and change the path request options so that it is continuously computed and checked.

10.7. Additional Optimization Options

To this point the connectivity matrix has been described as n x n, with each possible endpoint, e.g. a data center or customer network, being represented. In other words, there are n possible endpoints. For any pair of endpoints, if there is some connectivity between them, even if across multiple physical networks and domains, one or more paths between them could be continuously computed and stored.

However, this is probably unnecessary. Based on data center sites, physical and virtual connectivity, available functions, and customer interests, the sites that might connect with other sites are likely a significant subset of the full n x n possible connections. Therefore, it would make sense to weight the site-2-site connections.

A very simple method is to compute paths only for site-to-site pairs that commonly connect as part of network services. Other site-to-site paths would remain empty. There are likely a number of ways to improve this based upon other factors, including learned behavior. One option is to initially compute a full n x n connectivity matrix, with mutiple paths for different priorities. Subsequent refreshing of any connection depends upon the frequency of the use of that connection. The frequency of updates would depend upon the frequency of the connection use. Lower frequency connections will be assigned a lower weight. The weight is refreshed whenever the connection is used, with the weight raised and the ’time last assigned’ updated. On the other hand, another agent will execute periodically, and it will reduce each weight by a certain amount (though no lower than zero.)

When a new service is to be placed, if the orchestrator finds a lower weight connection to be useful, it might choose to use it. If the age is older than some configured threshold, it might ask the PCE to re-compute the path before it uses it.

10.8. Cached Paths Over Multiple Sites

In addition to the pre-computed site-2-site paths, another optimization step is to cache full or partial multi-site or multi data center paths. Is is likely there will be a fixed number of network services and each with a fixed, or mostly fixed, order of network functions. Even if some network services are dynamic in order, size and content (NFs), these will likely be a small minority, at the outset of NFV based network service delivery. Therefore, it will be possible to identify a finite number of network services, including their VNF-FGs, i.e. the ordering of the NFs, and needed compute and network resources. The frequencies and instantiation patterns of these services and their locations can be used to drive how often and how many cached instances should be computed, in order to ensure resources are available. Computing for multiple links in a network service also provides the opportunity to sychronize the computations of available resources. This should avoid the possibility of any connections in a network service unintentionally relying on the availability of bandwidth on the same physical link.

10.9. Summary

The purpose of this section has been to present options for computing optimized network service deployment that might drive requirements for a network SDN controller or PCE. The organization of this section has been intended to build and make clear how the use of pre-computed and cached paths between the data center sites can benefit in the delivery of optimzed services. It has described the creation and management of computing multiple paths for each pairwise set of data centers, based on different weights of factors, e.g. latency, bandwidth, etc. It also has outlined how pre-computed full service paths might also work. The result should be a greater likelihood of being able to deliver a service, as well as a significant reduction in the time to respond to service requests.

11. Connectivity based on Potential vs. Actual Topologies

Something missing here is how the orchestrator and PCE should consider ’potential connectivity’ as part of the VNF placement and connectivity process. For example, in the HPC use case, we know there will likely be a need for a high bandwidth low latency connection. This is probably something that is not available in the connectivity matrix as described so far. The matrix includes only information from the existing network topology configuration, including the configuration of existing tunnels, and performance metrics.

This calls out that there is a need for an agent providing connectivity information, e.g. PCE or SDN controller, to also present potential connectivity, should it be needed. This is not a simply issue. For example, updating or introducing a new underlying optical network circuit will affect the topology of the packet network running on top of it. This means that existing paths between sites would no longer be valid.

An orchestrator's deployment or placement algorithm should have the option to request this sort of 'potential' connectivity for any pair of sites, and the PCE or network controller that is computing these paths should be able to provide this.

This functionality could be used for near term needs, perhaps dedicating that circuit for a single network service, such as the HPC service. In the long term, other advantages might be realized. For example, if the system is creating a large number of multi-hop connections between two data centers, the orchestrator or PCE may recognize this and propose a possible direct circuit connection to reduce costs and latency. This is an area for more analysis. It is something that will drive new information exchange between functional blocks, e.g. the PCE or SDN controller, via its NBI and perhaps SBI, if hierarchical architectures are used.

12. Virtual Networks Equivalency

To this point the description has been for a relatively simple network, with paths computed for different data center connections. Another perspective on this is to consider each data center and user endpoint as a node in the network, where these nodes are connected by one or more virtual links. This is a virtual network then, where there is an abstraction layer between the physical and virtual networks. Each entry in the connectivity matrix described above represents a link between any two data centers. There may be more than one link, as these multiple links represent the multiple entries per matrix cell, with each created based on different priorities e.g. bandwidth, latency, etc. In fact, the connectivity matrix described effectively describes this sort of virtual network. As noted earlier, the connectivity matrix may be sparse. Equivalently, the virtual network may not be show a link between every two nodes (data centers, etc.)

One thing to consider is the possibility of multiple virtual networks, where each supports a different customer or some type of independent domain. Multiple virtual networks (connectivity matrices) created, for each per customer or domain. There might even be layering. For example, a customer may use virtual networks (plus compute resources) from multiple providers, weave these together, and

then offer services to its own customers. This is equivalent to having specific connectivity matrices from each separate network and then combining these to create per customer connectivity matrices that combine information from the separate network matrices.

13. Implications for the Network Controllers (SDN / PCE) and Orchestrators

First, the network connectivity matrix (virtual network) must be continuously computed and the connection (link between sites) information passed to the NFV network service orchestrator as updates are generated. The computations must consider topology, current resource allocations, measured utilization, analytics, e.g. trending, baselines, patterns, and policies. Path computation must compute multiple paths concurrenty, for the same endpoints, but different priorities, policies, etc. The computations are not for one set of service parameters, but to identify for a link the available resources (typically bandwidth) for meeting certain service requirements.

A PCE path request should configure path re-computation and updates at a constant rate, e.g. every 5 minutes, as well as compute updates whenever an event triggers it. Such an event might be a change in bandwidth, or perhaps latency, due to allocation over a path currently in the connectivity matrix, or congestion due to traffic volumes. Each time the PCE computes a path it must do so with the latest information regarding connectivity, link state, reservations, and utilization and performance metrics.

Note, unfortunatelypre-computation can create other problems. For example, when computing a path, it is possible that the same physical (or virtual) link could be used in more than one NF-NF connection for a single network service. A conflict might exist resulting in inadequate resources, and this would not be recognized. For smaller services this might not be significant; however, for services that might require a lot of bandwidth, e.g. a large vCDN deployment, this might make a difference.

Finally, for the cached multi-site and connection solution, this will require support for requesting and then computing the hop by hop paths, synchronized, and caching them such that a full soluton can be delivered quickly.

14. Relationship to TEAS TE Topology and ACTN

In the IETF there is a lot of relevant ongoing related to enhancing the PCE capabilities, the PCE Protocol, and the introduction of the Abstraction and Control of Transport Networks (ACTN) framework. A

number of concepts described here are really relevant to transport network services (including MPLS, segment routing, and technologies other than optical), whther supporting NFV network services or not. Some of these are described in the TEAS WG TE Topology, currently in development. These include the description and definitions of network nodes, TE virtual links, and multiple layers of abstraction. PCE enhancements are defined to support this, and the ACTN framework address management and control from the service layer.

15. Areas of Future Evaluation

This is an early draft of this Internet draft. Its intention has been to highlight, through use cases, the need for co-ordinated optimization of network and compute resources for NFV based network services delivery and to expand upon some solution options that lead to possible new requirements of the network controller (SDN / PCE.) There are a number of topics raised here, or which are related, that require further study. These include:

- Testbed to evaluate effectiveness of pre-computation of connectivity and full network service paths.
- Evaluate requirements to support forecast or pre-sechedule network services.
- Determine any controller / PCE (PCEP) NBI / SBI messaging updates that might be needed
- Evaluate implications of non-ACID computations vs. deployments and methods to reduce risk.
- Centralized vs. Distributed computation of optimization
- Review work in other IETF and standardswork - ONF Transport, MEF, FORCES, ALTO, NFVRG, SDNRG, SFC WG, etc.

16. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

17. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

18. References

18.1. 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>.

18.2. Informative References

[I-D.ietf-pce-hierarchy-extensions] Zhang, F., Zhao, Q., Dios, O., Casellas, R., and D. King, "Extensions to Path Computation Element Communication Protocol (PCEP) for Hierarchical Path Computation Elements (PCE)", draft-ietf-pce-hierarchy-extensions-03 (work in progress), July 2016.

[I-D.ietf-pce-pcep-yang] Dhody, D., Hardwick, J., Beeram, V., and j. jefftant@gmail.com, "A YANG Data Model for Path Computation Element Communications Protocol (PCEP)", draft-ietf-pce-pcep-yang-01 (work in progress), October 2016.

[I-D.ietf-pce-stateful-pce] Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-17 (work in progress), November 2016.

[I-D.ietf-teas-actn-framework] Ceccarelli, D. and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework-01 (work in progress), October 2016.

[I-D.ietf-teas-pce-central-control] Farrel, A., Zhao, Q., Li, Z., and C. Zhou, "An Architecture for Use of PCE and PCEP in a Network with Central Control", draft-ietf-teas-pce-central-control-00 (work in progress), September 2016.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-05 (work in progress), October 2016.

[I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo-06 (work in progress), October 2016.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5441] Vasseur, JP., Ed., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, DOI 10.17487/RFC5441, April 2009, <http://www.rfc-editor.org/info/rfc5441>.

Author’s Address

Andrew Veitch (editor)
Netcracker, Inc.
95 Sawyer Road
Waltham 02453
US

Email: andrew.veitch@netcracker.com