

NFVRG
Internet-Draft
Intended status: Informational
Expires: January 16, 2019

P. Aranda Gutierrez
UC3M
D. Lopez
Telefonica
S. Salsano
Univ. of Rome Tor Vergata/CNIT
E. Batanero
July 15, 2018

High-level VNF Descriptors using NEMO
draft-aranda-nfvrg-recursive-vnf-06

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 January 16, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology and abbreviations	3
3. Prior art	3
3.1. Virtual network function descriptors	3
3.1.1. OpenMANO VNFDs	4
3.1.2. ETSI MANO VNFDs	5
3.2. NEMO	8
4. Additional requirements on NEMO	9
4.1. Referencing VNFDs in a NodeModel	9
4.2. Referencing the network interfaces of a VNF in a NodeModel	9
4.3. An example	9
5. Implementation	10
6. Operational Experience	11
7. Future work	13
8. Conclusion	13
9. IANA Considerations	13
10. Security Considerations	13
11. Acknowledgement	13
12. References	13
12.1. Normative References	13
12.2. Informative References	14
12.3. URIs	14
Authors' Addresses	15

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

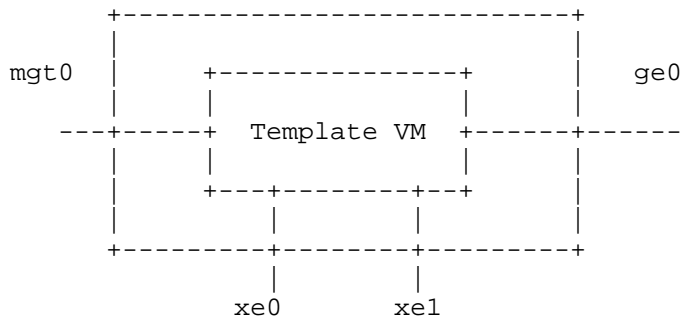
Virtual network function descriptors (VNFDs) are used in the Management and orchestration (MANO) framework of the ETSI NFV to achieve the optimal deployment of virtual network functions (VNFs). The Virtual Infrastructure Manager (VIM) uses this information to place the functions optimally. VNFDs include 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 VNFDs
  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 VNFDs is expressed by using the abstraction of Virtual Links, which interconnect Connection Points of the VNFDs. 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.

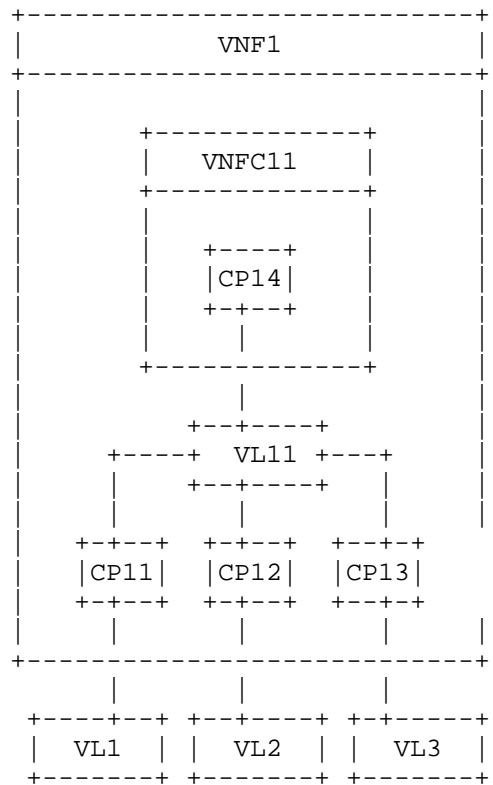


Figure 2: VNF example

```
#####
# 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: vdull1
  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 to 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 possible 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/nfvlab  
/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 \
    Endnodes 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:

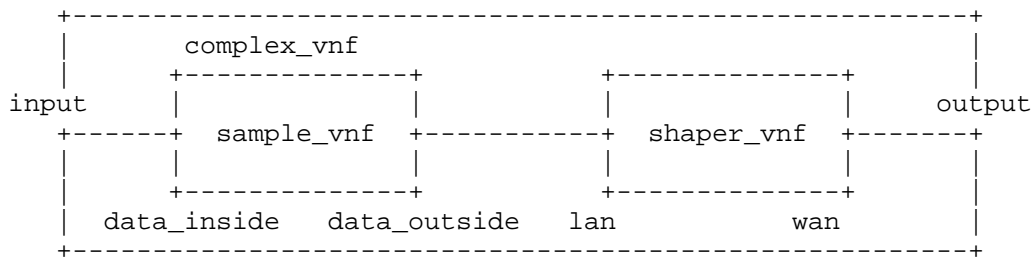


Figure 6

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 in 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 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.

This implementation is currently being updated to OpenDaylight Oxygen (the latest version at the time of writing), as a first step towards an ODL-independent implementation.

6. Operational Experience

We have used NEMO descriptors in the context of the MAMI Project [7], to describe a measurement network service based on three virtual network function components:

1. A Traffic [8] traffic generator and sink based on iperf3.
2. A tshark [9]-based packet capture
3. An InfluxDB [10]-based time series database to store measurements

The Network Service Descriptor must always include two instances of the traffic-based VNFC, while the tshark VNFC and the influxdb VNFC are optional (more information is provided at the traffic VNFC creation description page [11].)

The process of creating the different node models is incremental. We start by importing the node models:

```
CREATE NodeModel traffic VNFD https://<repo_url>/traffic.yaml;  
    ConnectionPoint mgmt at VNFD:eth0;  
    ConnectionPoint gen at VNFD:eth1;  
  
CREATE NodeModel tshark VNFD https://<repo_url>/tshark.yaml;  
    ConnectionPoint mgmt at VNFD:eth0;  
    ConnectionPoint probe at VNFD:eth1;  
  
CREATE NodeModel influxdb VNFD https://<repo_url>/influxdb.yaml;  
    ConnectionPoint mgmt at VNFD:eth0;
```

Figure 7: Creating VNFCs

Then, we create the kernel NSD, based on the traffic VNFCs only:

```

CREATE NodeModel traffic_kernel;
  Node iperf-servers Type traffic;
  Node iperf-clients Type traffic;
  ConnectionPoint client;
  ConnectionPoint server;
  ConnectionPoint mgmt;
  Connection icon Type p2p Endnodes client, iperfs-clients:gen;
  Connection ocon Type p2p Endnodes server, iperfs-servers:gen;
  Connection mgmt Type Lan \
    Endnodes mgmt, iperfs-servers:mgmt, iperfs-clients:mgmt;

```

Figure 8: Kernel NSD based on the traffic VNFCs

Adding the influxdb VNFC to create an autonomous measurement NSD that includes local storage for the measurement results is accomplished with the following NEMO script:

```

CREATE NodeModel
  Node iperf-servers Type traffic_kernel;
  Node database Type influxdb;
  ConnectionPoint client;
  ConnectionPoint server;
  ConnectionPoint mgmt;
  Connection icon Type p2p Endnodes client, traffic_kernel:client;
  Connection ocon Type p2p Endnodes server, traffic_kernel:server;
  Connection mgmt Type Lan \
    Endnodes mgmt, traffic_kernel:mgmt, influxdb:mgmt;

```

Figure 9: Adding influxdb

NEMO has shown a fundamental advantage when compared to YAML or JSON-based descriptors: since it is human-understandable, the development and debugging times of moderate to complex network service descriptors have been shortened considerably and the learning curve is much shallower compared with the original formats.

NEMO allows to identify requirements both for itself and MANO developers more quickly. An example is the connection of the wireshark-based traffic sniffing VNFC. The current connection types (LAN or p2p) do not consider port mirroring, a functionality provided by the TAPaaS plugin in Openstack. This requirement will be fed back to the different MANO communities (OSM, etc.) as a user requirement.

7. Future work

Future work includes extensions to the language to separate control and data plane connections explicitly and new types of connectivity models, including a model that provides the TAP as a Service [12] (TAPaaS) functionality available for OpenStack.

8. 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 and OSM descriptor formats in this document and for the reference implementation, other descriptors and concepts (i.e. as those used by TOSCA [13]) can also be used as the lowest level in this extension to the NEMO language.

9. IANA Considerations

This draft includes no request to IANA.

10. 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.

11. Acknowledgement

The work presented in this paper is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.

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, <<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.

12.2. 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.

12.3. 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] <https://mami-project.eu>
- [8] <https://github.com/mami-project/traffic/>
- [9] <https://www.wireshark.org>
- [10] <https://www.influxdata.com>
- [11] <https://github.com/mami-project/traffic/blob/master/README-VM.md>
- [12] https://docs.openstack.org/developer/dragonflow/specs/tap_as_a_service.html
- [13] <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: Experimental
Expires: September 6, 2018

CJ. Bernardos
UC3M
A. Mourad
InterDigital
March 5, 2018

IPv6-based discovery and association of Virtualization Infrastructure
Manager (VIM) and Network Function Virtualization Orchestrator (NFVO)
draft-bernardos-nfvrg-vim-discovery-00

Abstract

Virtualized resources do not need to be limited to those available in traditional data centers, where the infrastructure is stable, static, typically homogeneous and managed by a single admin entity. Computational capabilities are becoming more and more ubiquitous, with terminal devices getting extremely powerful, as well as other types of devices that are close to the end users at the edge (e.g., vehicular onboard devices for infotainment, micro data centers deployed at the edge, etc.). It is envisioned that these devices would be able to offer storage, computing and networking resources to nearby network infrastructure, devices and things (the fog paradigm). These resources can be used to host functions, for example to offload/complement other resources available at traditional data centers, but also to reduce the end-to-end latency or to provide access to specialized information (e.g., context available at the edge) or hardware.

This document describes mechanisms allowing dynamic discovery of virtualization resources and orchestrators in IPv6-based networks. New IPv6 neighbor discovery options are 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 <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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Network Function Virtualization	4
4. Fog Virtualization Overview	7
5. Problem statement	8
6. Advertisement and discovery of mobile resources (VIM+NFVI)	9
6.1. IPv6 ND-based discovery	11
7. IANA Considerations	11
8. Security Considerations	11
9. Acknowledgments	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Authors' Addresses	12

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 data centers. These services have different characteristics if compared with conventional IT services that have to be taken into account in this cloudification process. Also the transport network is affected in that it is

evolving to a more sophisticated form of IP architecture with trends like separation of control and data plane traffic, and more fine-grained forwarding of packets (beyond looking at the destination IP address) in the network to fulfill new business and service goals.

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

Network function virtualization (NFV) [etsi_nfv_whitepaper] and software defined networking (SDN) [onf_sdn_architecture] are changing the way the telecommunications sector will deploy, extend and operate their networks. The ETSI NFV Industry Specification Group (ISG) is developing the baseline NFV architecture, under some assumptions to make this development easier. One of these assumptions is that the resources used to run the virtualized functions are well known in advance by the management and orchestration entities, as well as stable. This document goes beyond this assumption [I-D.irtf-nfvrg-gaps-network-virtualization], by describing mechanisms allowing dynamic discovery of virtualization resources and orchestrators in IPv6-based networks.

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].

While [RFC2119] describes interpretations of these key words in terms of protocol specifications and implementations, they are used in this

document to describe requirements for the SFC mechanisms to efficiently enable fog RAN.

The following terms used in this document are defined by the ETSI NFV ISG, 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.

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 Virtualisation Infrastructure (NFVI).

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

3. 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):

- o Virtualized Network Function, running over the NFVI.

- o NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualized. NFVI supports the execution of the VNFs.
- o 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.

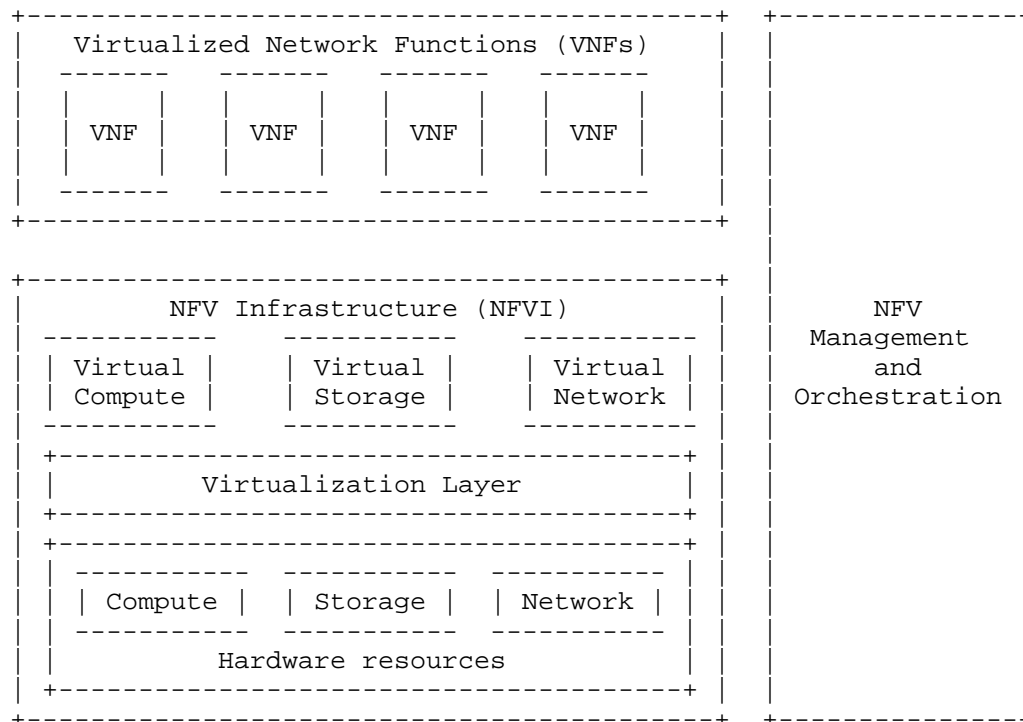


Figure 1: ETSI NFV framework

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):

- o Virtualized Network Function (VNF).
- o Element Management (EM).

- o NFV Infrastructure, including: Hardware and virtualized resources, and Virtualization Layer.
- o 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).

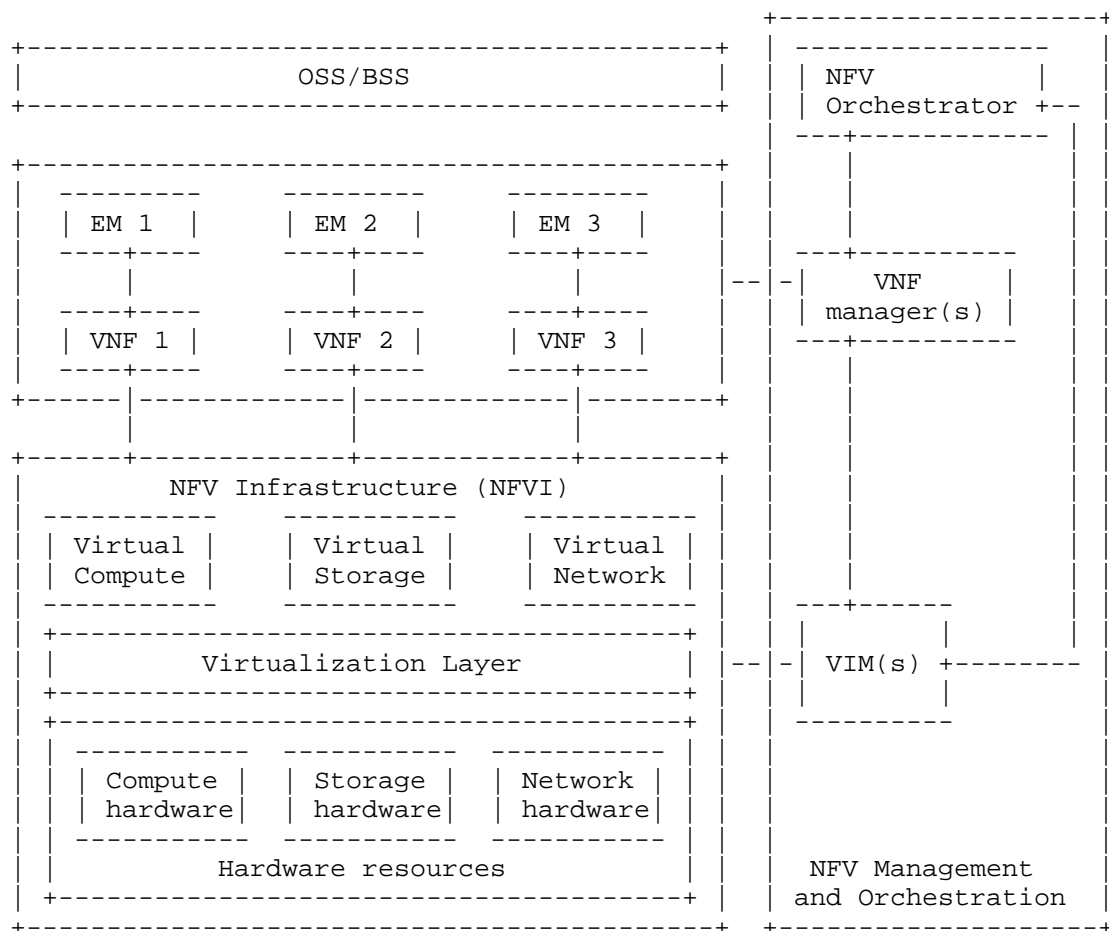


Figure 2: ETSI NFV reference architecture

4. Fog Virtualization Overview

Virtualization is invading all domains of the E2E 5G network, including the access, as a mean to achieve the necessary flexibility in support of the E2E slicing concept. The ETSI NFV framework is the cornerstone for making virtualization such a promising technology that can be matured in time for 5G. Typically, virtualization has been mostly envisaged in the core network, where sophisticated data centers and clouds provided the right substrate. And mostly, the framework focused on virtualizing network functions, so called VNFs (virtualized network functions), which were somewhat limited to functions that are delay tolerant, typically from the core and aggregation transport.

As the community has recently been developing the 5G applications and their technical requirements, it has become clear that certain applications would require very low latency which is extremely challenging and stressing for the network to deliver through a pure centralized architecture. The need to provide networking, computing, and storage capabilities closer to the users has therefore emerged, leading to what is known today as the concept of intelligent edge. ETSI has been the first to address this need recently by developing the framework of mobile edge computing (MEC).

Such an intelligent edge could not be envisaged without virtualization. Beyond applications, it raises a clear opportunity for networking functions to execute at the edge benefiting from inherent low latencies.

Whilst it is appreciated the particular challenge for the intelligent edge concept in dealing with mobile users, the edge virtualization substrate has been largely assumed to be fixed or stationary. Although little developed, the intelligent edge concept is being extended further to scenarios where for example the edge computing substrate is on the move, e.g., on-board a car or a train, or that it is distributed further down the edge, even integrating resources from different stakeholders, into what is known as the fog. The challenges and opportunities for such extensions of the intelligent edge remain an exciting area of future research.

Figure 3 shows a diagram representing the fog virtualization concept. The fog is composed by virtual resources on top of heterogeneous resources available at the edge and even further in the RAN and end-user devices. These resources are therefore owned by different stakeholders who collaboratively form a single hosting environment for the VNFs to run. As an example, virtual resources provided to the fog might be running on eNBs, APs, at micro data centers deployed in shopping malls, cars, trains, etc. The fog is connected to data

centers deeper into the network architecture (at the edge or the core). On the top part of the figure, an example of user and control plane VNFs is shown. User plane VNFs are represented as "fx", and control ones as "ctrlx". Depending on the functionality implemented by these VNFs and the service requirements, these VNFs would be mapped (i.e., instantiated) differently to the physical resources (as described in [I-D.aranda-sfc-dp-mobile]).

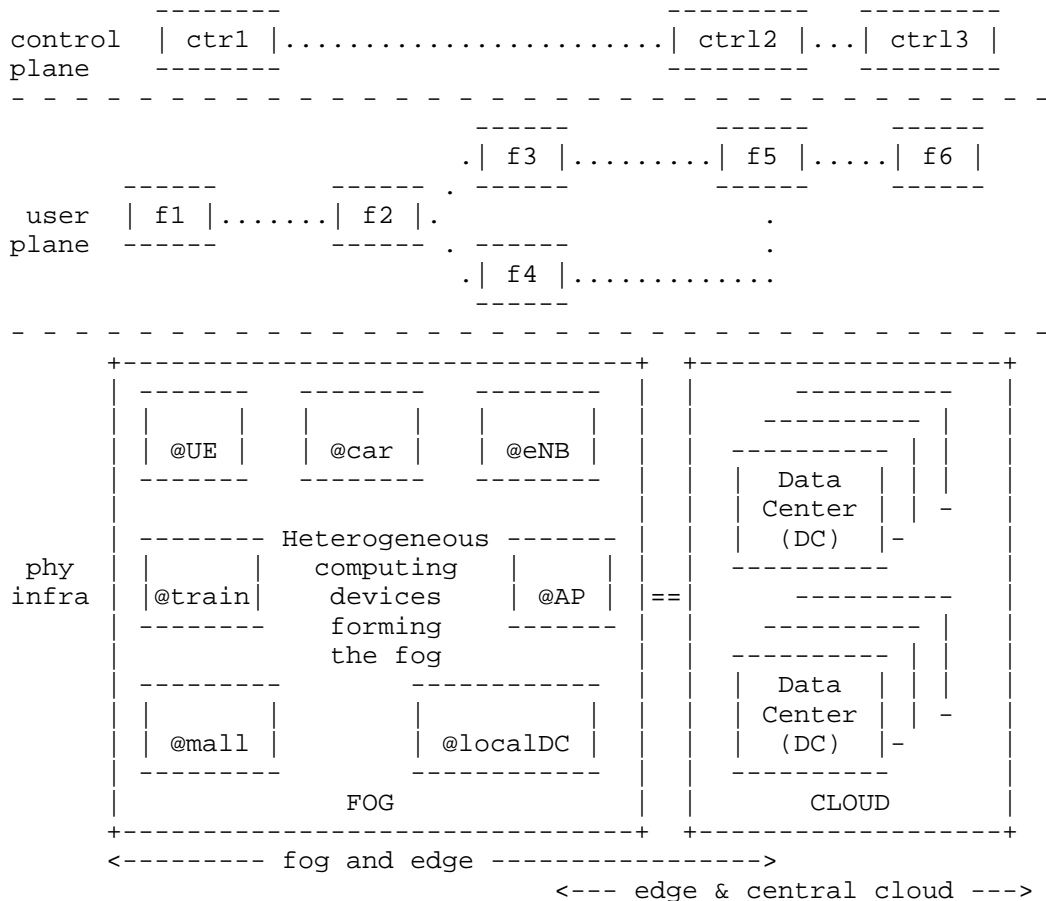


Figure 3: Fog virtualization

5. Problem statement

Virtualized resources do not need to be limited to those available in traditional data centers, where the infrastructure is stable, static, typically homogeneous and managed by a single admin entity. Computational capabilities are becoming more and more ubiquitous,

with terminal devices getting extremely powerful, as well as other types of devices that are close to the end users at the edge (e.g., vehicular onboard devices for infotainment, micro data centers deployed at the edge, etc.). It is envisioned that these devices would be able to offer storage, computing and networking resources to nearby network infrastructure, devices and things (the fog paradigm). These resources can be used to host functions, for example to offload/complement other resources available at traditional data centers, but also to reduce the end-to-end latency or to provide access to specialized information (e.g., context available at the edge) or hardware.

Since the fog resources are volatile, i.e. may dynamically appear and disappear, and may be mobile, i.e. may move from one place to another, mechanisms to discover and advertise virtualized fog resources are required.

Taking ETSI NFV architecture (see Section 3) as a baseline for the virtualization of the fog nodes, the discovery of a virtualization resource can be done either through (i) the discovery of NFVI from a VIM; or through (ii) the discovery of VIMs and associated NFVI from an NFVO. In this document we focus on the alternative ii) that is the discovery of the VIMs and NFVI from an NFVO. This is so because a VIM is typically NFVI-specific, and therefore these two are more often than not tied together.

The relationship between an NFVO and the resources it is capable to orchestrate through a VIM is statically defined according to the current ETSI NFV specifications [etsi_nfv_002] [etsi_nfv_ifa_005]. The interface Or-Vi (between NFVO and VIM) [etsi_nfv_ifa_005] does not include any discovery and automatic registration of (mobile) VIMs from a (mobile) NFVO.

6. Advertisement and discovery of mobile resources (VIM+NFVI)

This document describes IPv6 extensions to allow discovery of virtualization resources, in the form of a VIM + associated NFVI. Examples of scenarios where this is useful are shown in Figure 4 and Figure 5, including also a high-level view of the solution.

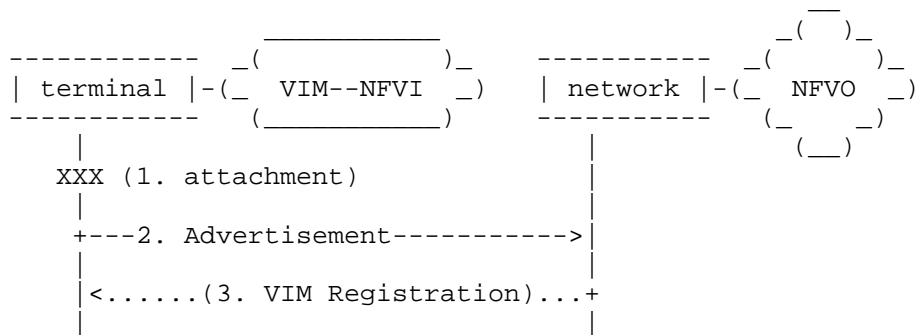


Figure 4: VIM+NFVI advertisement

Figure 4 shows an scenario in which a mobile terminal with available resources (NFVI, and associated VIM) attaches to a network (step 1). Then, it advertises (step 2) that it has virtualization resources (and their characteristics, such as the type of VIM) that could be eventually used. An NFVO sitting in the network can then decide to register the VIM for later use (step 3). This document specifies some options for step 2 based on IP signaling. Step 3 is implementation dependent and very much VIM-NFVO specific.

Similarly, Figure 5 shows a scenario with a mobile NFVO. A mobile terminal with an embedded NFVO attaches to a network (step 1). Then, it queries the network (step 2) to learn if there are virtualization resources available. If so, the network conveys that information (step 3). The NFVO can then decide to register the VIM for later use (step 4). This document specifies some options for steps 2 and 3 based on IP signaling. Step 4 is implementation dependent and very much VIM-NFVO specific.

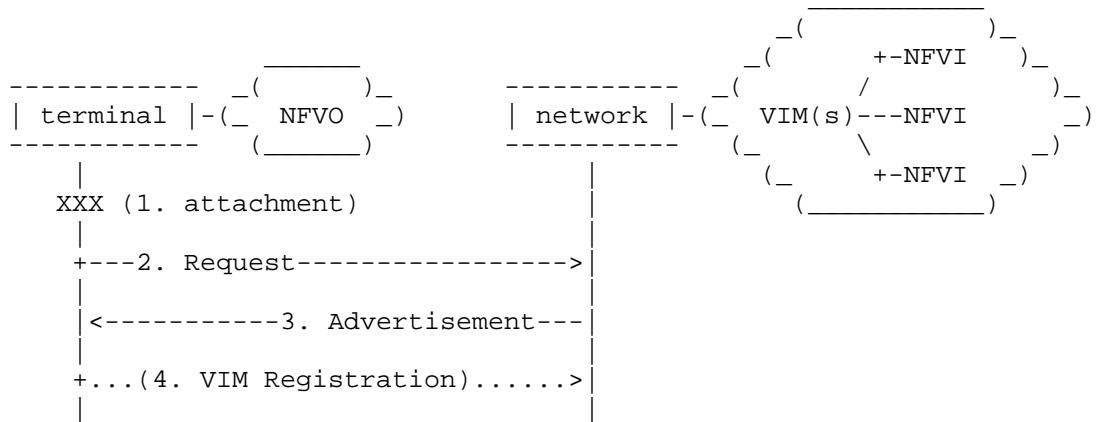


Figure 5: VIM+NFVI discovery

6.1. IPv6 ND-based discovery

TBD.

7. IANA Considerations

N/A.

8. Security Considerations

TBD.

9. Acknowledgments

The work in this draft will be further developed and explored under the framework of the H2020 5G-CORAL project (Grant 761586).

10. References

10.1. 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>>.

10.2. Informative References

- [etsi_nfv_002]
"Network Functions Virtualization (NFV); Architectural Framework," ETSI GS NFV 002 v1.1.1", October 2013.
- [etsi_nfv_ifa_005]
"Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification," ETSI GS NFV-IFA 005 V2.3.1", August 2017.
- [etsi_nfv_whitepaper]
"Network Functions Virtualisation (NFV). White Paper 2", October 2014.
- [I-D.aranda-sfc-dp-mobile]
Gutierrez, P., Gramaglia, M., Lopez, D., and W. Haeffner, "Service Function Chaining Dataplane Elements in Mobile Networks", draft-aranda-sfc-dp-mobile-04 (work in progress), June 2017.
- [I-D.irtf-nfvrg-gaps-network-virtualization]
Bernardos, C., Rahman, A., Zuniga, J., Contreras, L., Aranda, P., and P. Lynch, "Network Virtualization Research Challenges", draft-irtf-nfvrg-gaps-network-virtualization-09 (work in progress), February 2018.
- [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.
- [onf_sdn_architecture]
"SDN Architecture (Issue 1.1), ONF TR-521", February 2016.

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/>

Alain Mourad
InterDigital Europe

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

NFVRG
Internet-Draft
Intended status: Informational
Expires: March 6, 2019

CJ. Bernardos
UC3M
A. Rahman
InterDigital
JC. Zuniga
SIGFOX
LM. Contreras
TID
P. Aranda
UC3M
P. Lynch
Ixia
September 2, 2018

Network Virtualization Research Challenges
draft-irtf-nfvrg-gaps-network-virtualization-10

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).

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 March 6, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and scope	3
2. Terminology	4
3. Background	5
3.1. Network Function Virtualization	5
3.2. Software Defined Networking	7
3.3. ITU-T functional architecture of SDN	12
3.4. Multi-access Edge Computing	13
3.5. IEEE 802.1CF (OmniRAN)	14
3.6. Distributed Management Task Force	14
3.7. Open Source initiatives	14
4. Network Virtualization Challenges	16
4.1. Introduction	16
4.2. Guaranteeing quality-of-service	16
4.2.1. Virtualization Technologies	17
4.2.2. Metrics for NFV characterization	17
4.2.3. Predictive analysis	18
4.2.4. Portability	19
4.3. Performance improvement	19
4.3.1. Energy Efficiency	19

4.3.2. Improved link usage	20
4.4. Multiple Domains	20
4.5. 5G and Network Slicing	21
4.5.1. Virtual Network Operators	22
4.5.2. Extending Virtual Networks and Systems to the Internet of Things	23
4.6. Service Composition	24
4.7. End-user device virtualization	25
4.8. Security and Privacy	26
4.9. Separation of control concerns	27
4.10. Network Function placement	28
4.11. Testing	28
4.11.1. Changes in methodology	28
4.11.2. New functionality	30
4.11.3. Opportunities	31
5. Technology Gaps and Potential IETF Efforts	31
6. NFVRG focus areas	32
7. IANA Considerations	33
8. Security Considerations	33
9. Acknowledgments	33
10. Informative References	34
Authors' Addresses	39

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 (NFV) 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):

- o Virtualized Network Function, running over the NFVI.
- o NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualized. NFVI supports the execution of the VNFs.
- o 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.

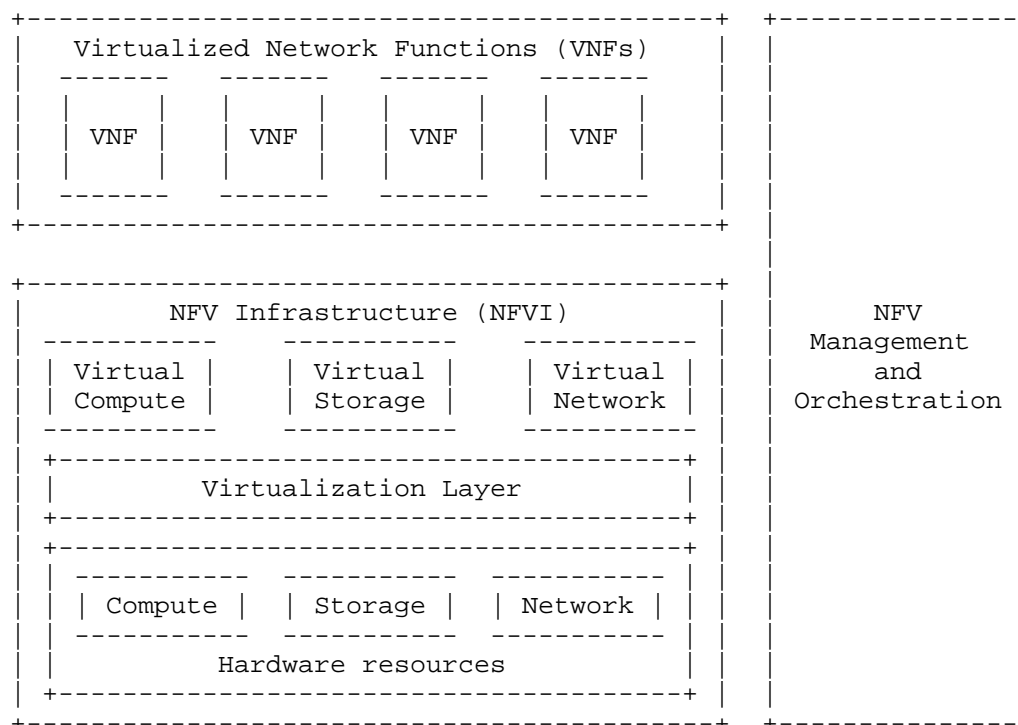


Figure 1: ETSI NFV framework

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):

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

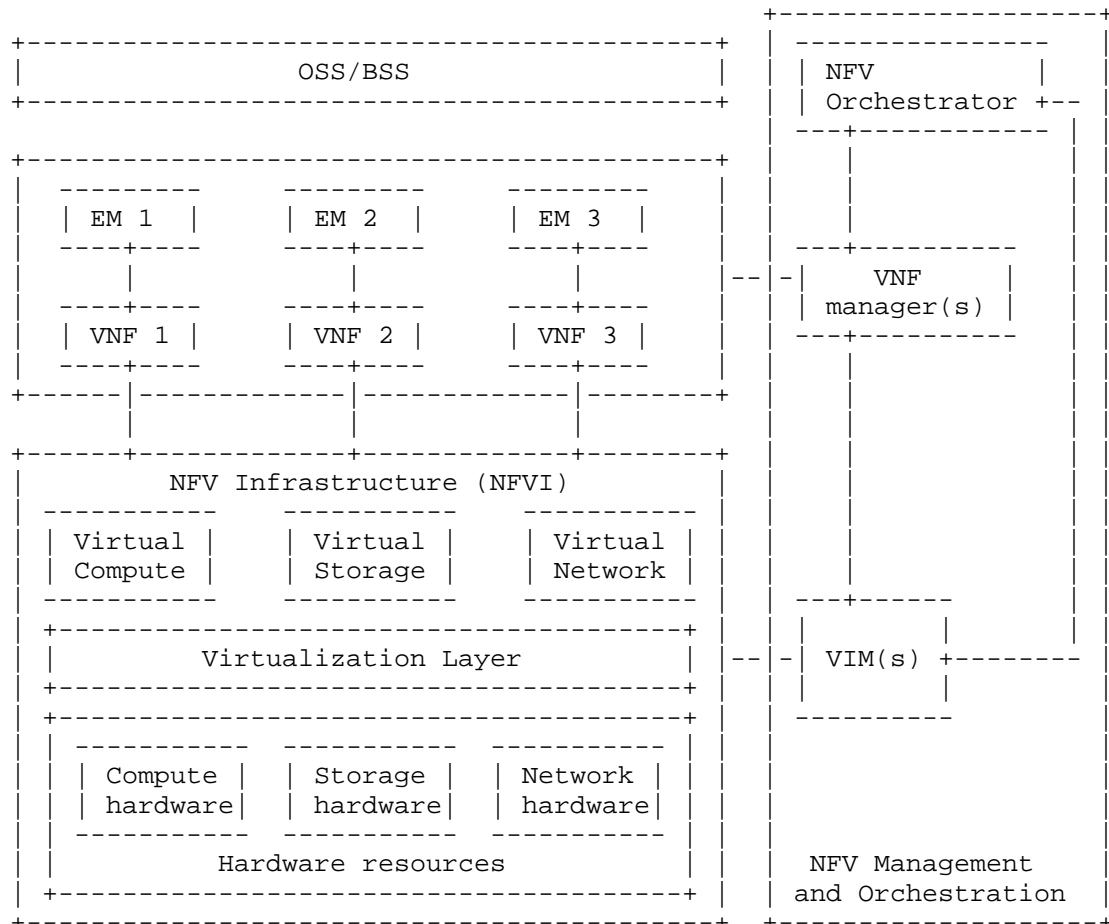


Figure 2: ETSI NFV reference architecture

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.

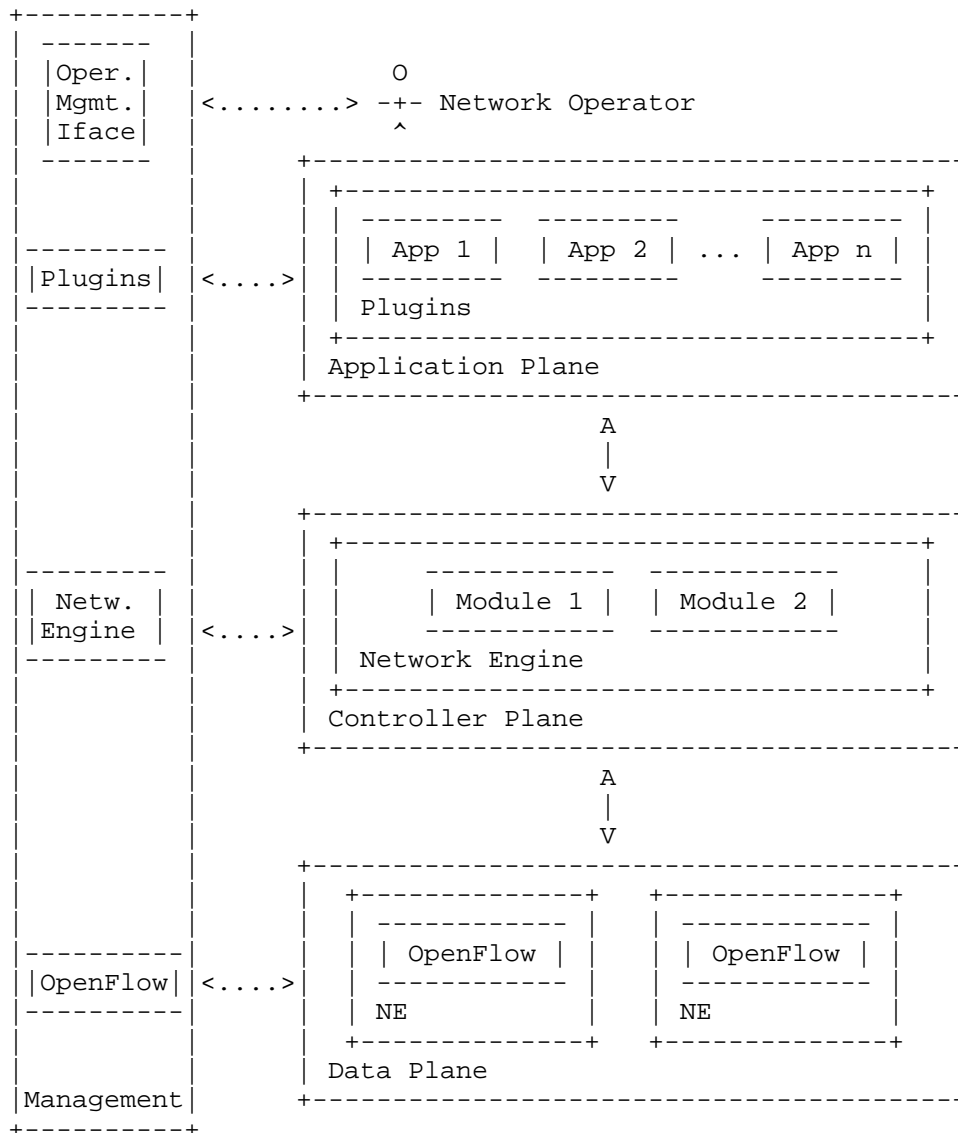


Figure 3: High level SDN ONF architecture

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).

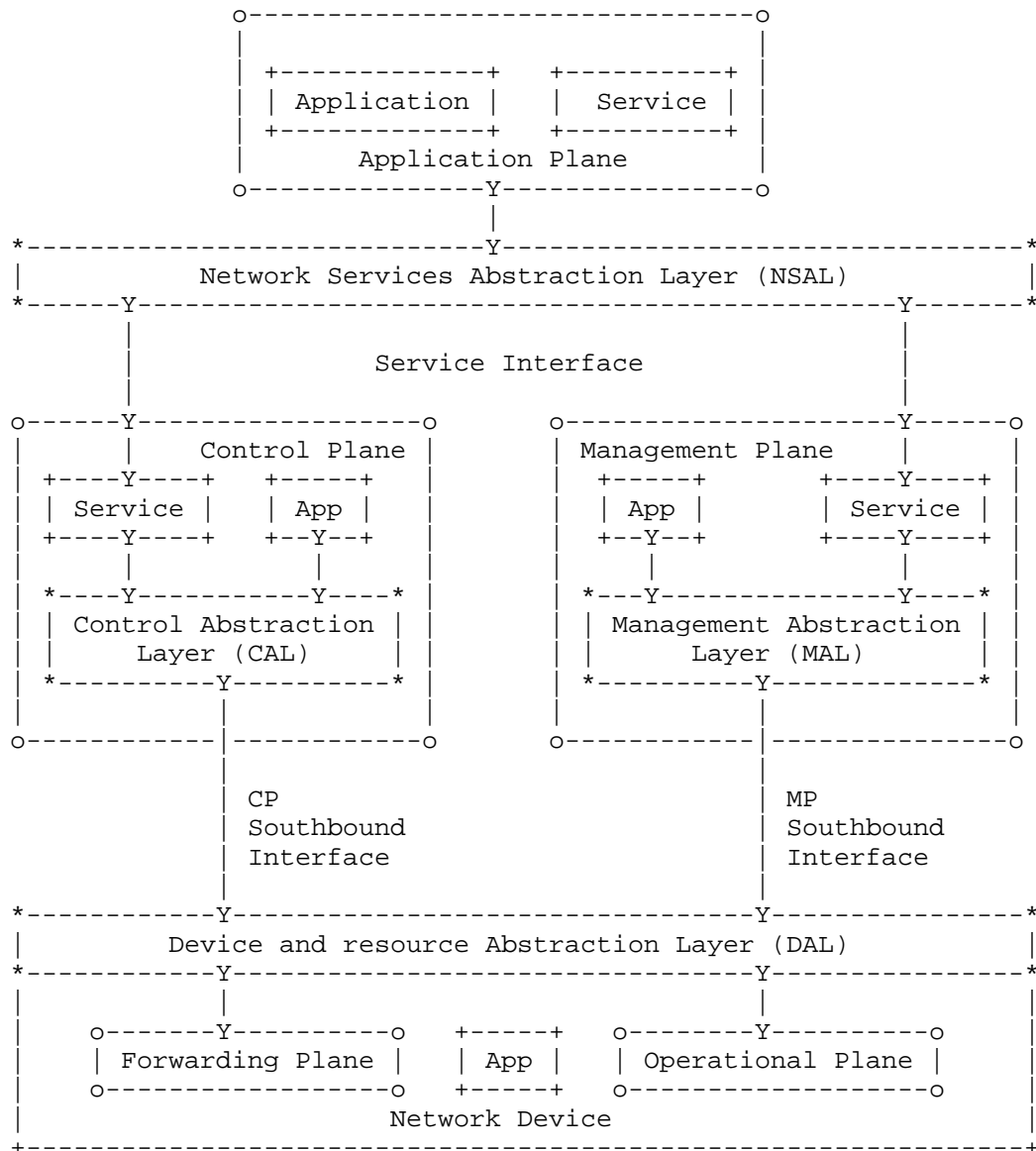


Figure 4: SDN Layer Architecture

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.

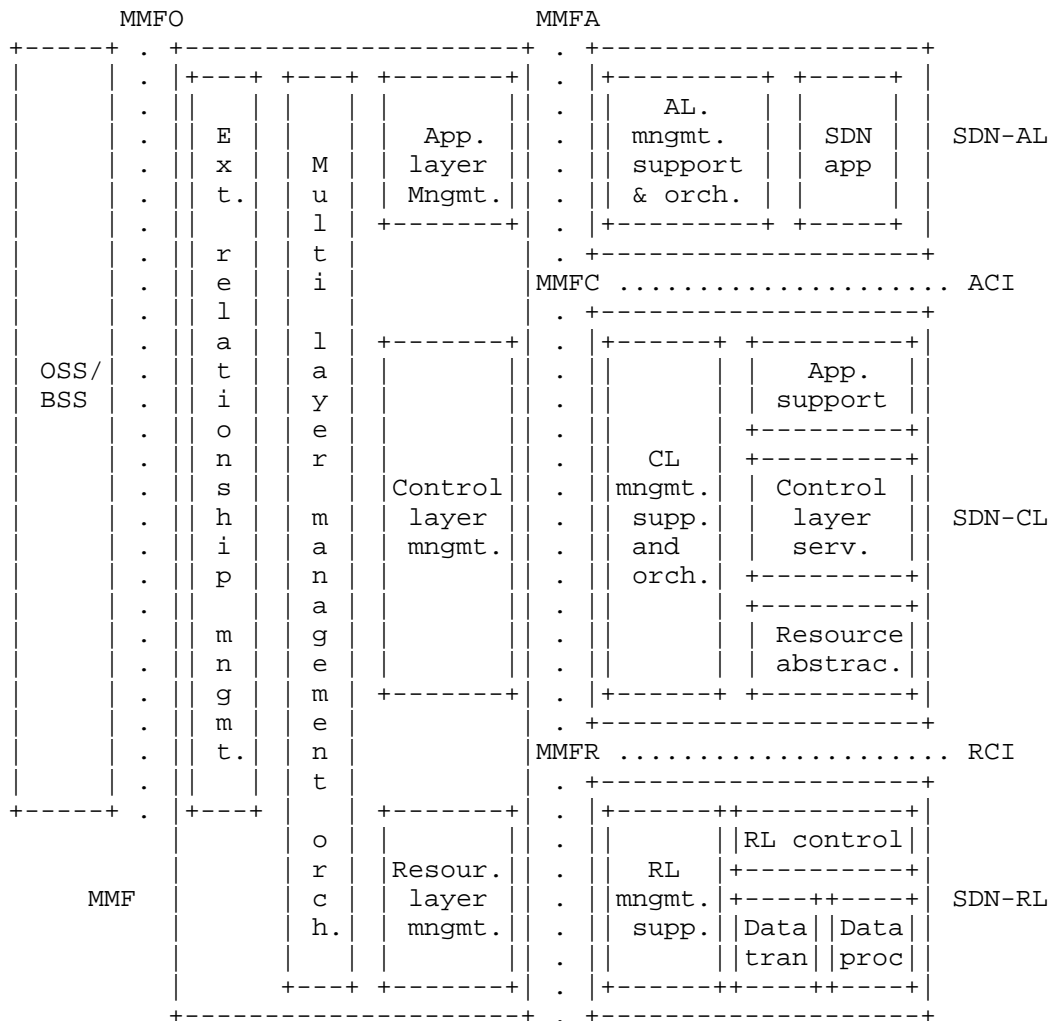


Figure 5: ITU-T SDN functional architecture

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:

- o 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.

- o 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.
- o 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.
- o 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.
- o 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.
- o 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.
- o 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 as Telefonica's OpenMANO or Canonical's Juju, among others.

- o 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.
- o 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.
- o 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]:

- o Energy aware scaling (e.g., reductions in CPU speeds and partially turning off some hardware components to meet a given energy consumption target).
- o Energy-aware function placement.
- o 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:

- o 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).
- o In any performance test bed, the test system should not share the same resources as the System Under Test (SUT). While multi-tenancy 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:

- o 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.
- o New management frameworks, considering aspects related to new OAM mechanisms (e.g., configuration control, hybrid descriptors) and lightweight MANO proposals.
- o Techniques to guarantee low latency, resource isolation, and other dataplane features, including hardware acceleration, functional offloading to dataplane elements (including NICs), and related approaches.
- o Measurement and benchmarking, addressing both internal measurements and external applications.

NFVRG Focus Point	Open Research Area
1-Re-architecting functions	<ul style="list-style-type: none"> - Performance improvem. - Network Slicing - Guaranteeing QoS - Security - End-user device virt. - Separation of control
2-New management frameworks	<ul style="list-style-type: none"> - Multiple Domains - Service Composition - End-user device virt.
3-Low latency, resource isolation, etc	<ul style="list-style-type: none"> - Performance improvem. - Separation of control
4-Measurement and benchmarking	<ul style="list-style-type: none"> - 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., Szabo, R., Li, X., Paolucci, F., Sgambelluri, A., Martini, B., Valcarengi, L., Landi, G., Andrushko, D., and A. Mourad, "Multi-domain Network Virtualization", draft-bernardos-nfvrg-multidomain-04 (work in progress), March 2018.

[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]

Geng, L., Slawomir, S., Qiang, L., kiran.makhijani@huawei.com, k., Galis, A., and L. Contreras, "Problem Statement of Common Operation and Management of Network Slicing", draft-geng-coms-problem-statement-04 (work in progress), March 2018.

[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
Labège 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