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

Copyright Notice

Table of Contents

1. Introduction

Currently, there is a lot of on-going activity to deploy NFV in the network. From the point of view of the orchestration, Virtual Network Functions are blocks that are deployed in the infrastructure as independent units. Following the reference architectural model proposed in [ETSI-NFV-MANO], VNFs provide for one layer of components (VNF components(VNFCs)) below, i.e. a set of VNFCs accessible to a VNF provider can be composed into VNFs. However, there is no simple way to use existing VNFs as components in VNFs with a higher degree of complexity. In addition, Network Service Descriptors (NSD) and VNF Descriptors (VNFDs) specified in [ETSI-NFV-MANO] and used in different open source MANO frameworks are YAML-based files, which despite being human readable, are not easy to understand.

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

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

2. Terminology and abbreviations

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

3. Prior art

3.1. Virtual network function descriptors

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

There are different MANO frameworks available. For this draft, we will first concentrate on the example of OpenMANO [2], which uses a

YAML [3] representation similar to the one specified in [ETSI-NFV-MANO]. Then we will provide an example using the exact format specified in [ETSI-NFV-MANO].

3.1.1. OpenMANO VNFDs

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

```
          +-----------------------------+
          |                             |
    mgt0  |     +----------------+      |   ge0
          |     |                |      |
       ---+-----+  Template VM   +------+-------
          |     |                |      |
          |     +---+--------+---+      |
          |         |        |          |
          +---------+--------+----------+
                    |        |
                   xe0      xe1
```

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

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

3.1.2. ETSI MANO VNFDs

In this example we consider the VNF represented in Figure 6.4 of [ETSI-NFV-MANO]. Its internal diagram, including a VNF component, is represented in Figure Figure 2. A YAML representation of the VNF Descriptor is reported in Figure Figure 3. The topology of the interconnection of VNFs is expressed by using the abstraction of Virtual Links, which interconnect Connection Points of the VNFs. The

Virtual Links are described by Virtual Link Descriptors (VLD) files. An example YAML representation of the Virtual Link VL1 in the example VNF is reported in Figure Figure 3. In order to understand the topology, a (potentially large) set of VNFD and VLD files needs to be analysed. For a human programmer of the service, this representation is not friendly to write and very hard to read/understand/debug.

```
+-----------------------------+
|            VNF1             |
+-----------------------------+
|                             |
|       +-------------+       |
|       |   VNFC11    |       |
|       +-------------+       |
|       |             |       |
|       |   +----+    |       |
|       |   |CP14|    |       |
|       |   +-+--+    |       |
|       |     |       |       |
|       +-------------+       |
|             |               |
|          +--+----+          |
|     +----+  VL11 +---+      |
|     |    +--+----+   |      |
|     |       |        |      |
|   +-+--+  +-+--+  +--+-+     |
|   |CP11|  |CP12|  |CP13|     |
|   +-+--+  +-+--+  +--+-+     |
|     |       |        |      |
+-----------------------------+
      |       |        |
 +----+--+ +--+----+ +-+-----+
 |  VL1  | |  VL2  | |  VL3  |
 +-------+ +-------+ +-------+
```

Figure 2: VNF example

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


Figure 3: ETSI MANO compliant VNF descriptor example

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

Figure 4: ETSI MANO compliant Virtual Link descriptor example

3.2. NEMO

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

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

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

Figure 5: Creating a NodeModel in NEMO

4. Additional requirements on NEMO

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

4.1. Referencing VNFDs in a NodeModel

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

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

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

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

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

4.3. An example

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

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

Import from a sample VNFD from the OpenMANO repository

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

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

```
    +---------------------------------------------------------+
    |       complex_vnf                                       |
    |      +--------------+          +--------------+        |
 input     |              |          |              |     output
    +------+  sample_vnf  +----------+  shaper_vnf  +--------+
    |      |              |          |              |        |
    |      +--------------+          +--------------+        |
    |  data_inside   data_outside   lan           wan        |
    +---------------------------------------------------------+
```

Figure 6

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

5. Implementation

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

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 Trafic [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 trafic-based VNFC, while the tshak VNFC and the influxdb VNFC are optional (more information is provided at the trafic VNFC creation description page [11].)

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

```
CREATE NodeModel trafic VNFD https://<repo_url>/trafic.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 trafic VNFCs only:

```
CREATE NodeModel trafic_kernel;
    Node iperf-servers Type trafic;
    Node iperf-clients Type trafic;
    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 trafic 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 trafic_kernel;
    Node database Type influxdb;
    ConnectionPoint client;
    ConnectionPoint server;
    ConnectionPoint mgmt;
    Connection icon Type p2p Endnodes client, trafic_kernel:client;
    Connection ocon Type p2p Endnodes server, trafic_kernel:server;
    Connection mgmt Type Lan \
        Endnodes mgmt, trafic_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/trafic/

[9] https://www.wireshark.org

[10] https://www.influxdata.com

[11] https://github.com/mami-project/trafic/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: Informational
Expires: March 7, 2019

CJ. Bernardos, Ed.
UC3M
LM. Contreras
TID
I. Vaishnavi
Huawei
R. Szabo
Ericsson
J. Mangues
CTTC
X. Li
NEC
F. Paolucci
A. Sgambelluri
B. Martini
L. Valcarenghi
SSSA
G. Landi
Nextworks
D. Andrushko
MIRANTIS
A. Mourad
InterDigital
September 3, 2018

Multi-domain Network Virtualization
draft-bernardos-nfvrg-multidomain-05

Abstract

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

Status of This Memo

Table of Contents

1. Introduction

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

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

Furthermore, 5G networks are being designed to be capable of fulfilling the needs of a plethora of vertical industries (e.g., automotive, eHealth, media), which have a wide variety of requirements [ngmn_5g_whitepaper]. The slicing concept tries to make the network of the provider aware of the business needs of tenants (e.g., vertical industries) by customizing the share of the network assigned to them. The term network slice was coined to refer to a complete logical network composed of network functions and the resources to run them [ngmn_slicing]. These resources include network, storage, and computing. The way in which services requested by customers of the provider are assigned to slices depends on customer needs and provider policies. The system must be flexible to accommodate a variety of options.

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

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

2. Terminology

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

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

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

NFV Orchestrator (NFVO): functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.

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

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

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

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

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

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

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

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

3. Background: the ETSI NFV architecture

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

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.

```
   +-------------------------------------------+  +---------------+
   |   Virtualized Network Functions (VNFs)    |  |               |
   |  -------   -------   -------   -------    |  |               |
   |  |     |   |     |   |     |   |     |    |  |               |
   |  | VNF |   | VNF |   | VNF |   | VNF |    |  |               |
   |  |     |   |     |   |     |   |     |    |  |               |
   |  -------   -------   -------   -------    |  |               |
   +-------------------------------------------+  |               |
                                                  |               |
   +-------------------------------------------+  |               |
   |         NFV Infrastructure (NFVI)         |  |      NFV      |
   | -----------    -----------    ----------- |  |  Management   |
   | | Virtual |    | Virtual |    | Virtual | |  |      and      |
   | | Compute |    | Storage |    | Network | |  | Orchestration |
   | -----------    -----------    ----------- |  |               |
   | +---------------------------------------+ |  |               |
   | |         Virtualization Layer          | |  |               |
   | +---------------------------------------+ |  |               |
   | +---------------------------------------+ |  |               |
   | | -----------  -----------  ----------- | |  |               |
   | | | Compute |  | Storage |  | Network | | |  |               |
   | | -----------  -----------  ----------- | |  |               |
   | |          Hardware resources           | |  |               |
   | +---------------------------------------+ |  |               |
   +-------------------------------------------+  +---------------+
```

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).

- Element Management (EM).

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

- Virtualized Infrastructure Manager(s) (VIM).

- NFV Orchestrator.

- VNF Manager(s).

- Service, VNF and Infrastructure Description.

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

```
                                                  +--------------------+
   +-------------------------------------------+  | ----------------   |
   |                 OSS/BSS                   |  | | NFV          |   |
   +-------------------------------------------+  | | Orchestrator +-- |
                                                  | ---+------------ | |
   +-------------------------------------------+  |    |             | |
   |  ---------     ---------     ---------    |  |    |             | |
   |  | EM 1  |     | EM 2  |     | EM 3  |    |  |    |             | |
   |  ----+----     ----+----     ----+----    |  | ---+----------   | |
   |      |             |             |        |--|-|    VNF     |   | |
   |  ----+----     ----+----     ----+----    |  | | manager(s) |   | |
   |  | VNF 1 |     | VNF 2 |     | VNF 3 |    |  | ---+----------   | |
   |  ----+----     ----+----     ----+----    |  |    |             | |
   +------|-------------|-------------|--------+  |    |             | |
          |             |             |           |    |             | |
   +------+-------------+-------------+--------+  |    |             | |
   |         NFV Infrastructure (NFVI)         |  |    |             | |
   | -----------    -----------    ----------- |  |    |             | |
   | | Virtual |    | Virtual |    | Virtual | |  |    |             | |
   | | Compute |    | Storage |    | Network | |  |    |             | |
   | -----------    -----------    ----------- |  | ---+------       | |
   | +---------------------------------------+ |  | |        |       | |
   | |         Virtualization Layer          | |--|-| VIM(s) +-------- |
   | +---------------------------------------+ |  | |        |         |
   | +---------------------------------------+ |  | ----------         |
   | | -----------  -----------  ----------- | |  |                    |
   | | | Compute |  | Storage |  | Network | | |  |                    |
   | | | hardware|  | hardware|  | hardware| | |  |                    |
   | | -----------  -----------  ----------- | |  |                    |
   | |          Hardware resources           | |  |  NFV Management    |
   | +---------------------------------------+ |  | and Orchestration  |
   +-------------------------------------------+  +--------------------+
```

Figure 2: ETSI NFV reference architecture

4. Multi-domain problem statement

Market fragmentation results from having a multitude of telecommunications network and cloud operators each with a footprint focused to a specific region. This makes it difficult to deploy cost effective infrastructure services, such as virtual connectivity or compute resources, spanning multiple countries as no single operator has a big enough footprint. Even if operators largely aim to provide the same infrastructure services (VPN connectivity, compute resources based on virtual machines and block storage), inter-operator collaboration tools for providing a service spanning several administrative boundaries are very limited and cumbersome. This makes service development and provisioning very time consuming. For example, having a VPN with end-points in several countries, in order to connect multiple sites of a business (such as a hotel chain), requires contacting several network operators. Such an approach is possible only with significant effort and integration work from the side of the business. This is not only slow, but also inefficient and expensive, since the business also needs to employ networking specialists to do the integration instead of focusing on its core business

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

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

5. Multi-domain architectural approaches

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

5.1. ETSI NFV approaches

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

The first option (shown in Figure 3) is based on a split of the NFVO into Network Service Orchestrator (NSO) and Resource Orchestrator (RO). A use case that this separation could enable is the following:

a network operator offering its infrastructure to different departments within the same operator, as well as to a different network operator like in cases of network sharing agreements. In this scenario, an administrative domain can be defined as one or more data centers and VIMs, providing an abstracted view of the resources hosted in it.

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

```
                             -------
                             | NSO |
                            /-------\
                           /         \
               --------   /  -------- \    --------
               | VNFM |   |  | VNFM |  |   | VNFM |
               --------  /   --------   \  --------
                  / _____/       /  \       \_____ \
                 / / __________/    \__________ \ \
                / / /                           \ \ \
   +-----------/-/-/---------+     +----------\-\-\----------+
   |        ---------        |     |        ---------        |
   |        |  RO   |        |     |        |  RO   |        |
   |        ---------        |     |        ---------        |
   |       /    |    \       |     |       /    |    \       |
   |      /     |     \      |     |      /     |     \      |
   |     /      |      \     |     |     /      |      \     |
   | ------- ------- ------- |     | ------- ------- ------- |
   | |VIM 1| |VIM 2| |VIM 3| |     | |VIM 1| |VIM 2| |VIM 3| |
   | ------- ------- ------- |     | ------- ------- ------- |
   | Administrative domain A |     | Administrative domain B |
   +-------------------------+     +-------------------------+
```

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

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

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

```
                           ------------
                           | Umbrella |
                           |   NFVO   |
                           ------------
                              / |  \
                            /   |    \
                           / -------- \
                          /  | VNFM |  \
                        /    --------    \
                       /        |         \
                     /       -------        \
                    /        |VIM 1|         \
                  /          -------           \
   --------------/------------     -------------\-------------
   |        --------         |     |        --------         |
   |        | NFVO |         |     |        | NFVO |         |
   |        --------         |     |        --------         |
   |          | | |          |     |          | | |          |
   | -------- | | | -------- |     | -------- | | | -------- |
   | | VNFM | | | | | VNFM | |     | | VNFM | | | | | VNFM | |
   | -------- | | | -------- |     | -------- | | | -------- |
   |   |  \__/__|__\_/_   |  |     |   |  \__/__|__\_/_   |  |
   |   |  __/___|___/\ \  |  |     |   |  __/___|___/\ \  |  |
   |   | / /    |     \ \ |  |     |   | / /    |     \ \ |  |
   | ------- ------- ------- |     | ------- ------- ------- |
   | |VIM 1| |VIM 2| |VIM 3| |     | |VIM 1| |VIM 2| |VIM 3| |
   | ------- ------- ------- |     | ------- ------- ------- |
   | Administrative domain A |     | Administrative domain B |
   +-------------------------+     +-------------------------+
```

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

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

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

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

```
+--------------------------------------------------+
|    NFVIaaS consumer's administrative domain       |
|                                                  |
|  +----------+                                    |
|  |   NS(s)  |                                    |
|  +----------+                                    |
|                                                  |
|  +----------+   +----------+   +----------+      |
|  |  VNF(s)  |   | VNFM(s)  |   |   NFVO   |      |
|  +----------+   +----------+   +----------+      |
|                                                  |
+-------------------------+------------------------+
                          +
 Administrative domain    +
++++++++++++++++++++++++++++++++++++++++++++++++++++
 boundary                 + NFVIaaS
                          +
+-------------------------+------------------------+
|                                                  |
|  +----------+   +-----------+                    |
|  |  NFVI    |   |   VIM(s)  |                    |
|  +----------+   +-----------+                    |
|                                                  |
+--------------------------------------------------+
```

Figure 5: NFVI use case

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

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

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

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

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

```
     +-------------------------------------------------+
     |    NFVIaaS consumer's administrative domain     |
     |                                                 |
     |  +-----------+               +------------+     |
     |  |   VNFM    |---+           |   NFVO     |     |
     |  +-+---------+   |---+       +-+--+-------+     |
     |    + +-+---------+   |        +   +             |
     |    +   + +----+----+-+       +    +             |
     |    +   +     +     +        +    +              |
     +----+---+----+------+-------+----+--------------+
          +   +    +      +      +    +
IFA 006 --+---+---+-------+--  --+----+-- IFA 005
          +   +  +        +     +    +
     +----+---+--+--------+----+----+-----------------+
     |    +   +  +   ++++++++++    +                   |
     |  +-+---+--+-+ +    +       +                    |
     |  |   VIM    |-+-+  +      +                     |
     |  +----------+   |--+-+   +                      |
     |      +----------+    ++++                       |
     |          |    VIM    |                          |
     |          +-----------+                          |
     |    NFVIaaS provider's administrative domain     |
     +-------------------------------------------------+
```

Figure 6: NFVIaaS architecture: MLPOC option

```
        +------------------------------------------------+
        |    NFVIaaS consumer’s administrative domain    |
        |                                                |
        |  +-----------+               +------------+    |
        |  |   VNFM    |---+           |   NFVO     |    |
        |  +-+---------+   |---+       +-+----------+    |
        |    + +-----------+   |        +                |
        |    +     |   VNFM    |        +                |
        |     +    +---------+-+       +                 |
        |      +             +        +                  |
        +-------+------------+-------+-------------------+
                 +           +      +
   IFA 006 -------+----------+--  --+-- IFA 005
                   +         +     +
        +-----------+--------+----+----------------------+
        |            +       +   +                       |
        |         +---+------+--+--+                     |
        |         | SLPOC function |                     |
        |         +-+---+---+------+                     |
        |          +    +   +                            |
        |      ---+-----+---+--- IFA 005                 |
        |        +      +   +                            |
        |  +----+-----+ +   +                            |
        |  |   VIM    |-+-+ +                            |
        |  +----------+   |-+-+                          |
        |      +----------+   |                          |
        |          |   VIM    |                          |
        |          +----------+                          |
        |    NFVIaaS provider’s administrative domain    |
        +------------------------------------------------+
```

Figure 7: NFVIaaS architecture: SLPOC option

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

```
                |
     **********+**************************************
     *          |                                     *
     *  +-------+------+                              *
     *  |              |                              *
   --+--+ Nested NS A  +-------+                      *
     *  |              |       +---------+            *
     *  +-------+------+                 |            *
     *          |                        |            *
     *          |              +-----+--------+      *
     *          |              |              |      *
     *          +-----------------+ Nested NS B  +-----+---
     *                         |              |      *
     *                         +-------+------+      *
     *        Composite NS C           |              *
     *                                 |              *
     **********************************+*************
                                       |
```

Figure 8: Composite and nested NSs

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

```
            +---------------------------------------------+
            |                                             |
            |                             +-------------+ |
            |                       +++++++   VNFM1-1   | |
            |    +-----------+     +      +-------------+ |
            |    |  NFVO-1   ++++++                       |
            |    +---+-------+     +      +-------------+ |
            |        +             +++++++   VNFM1-2   | |
            |        +                    +-------------+ |
            |        +    Administrative domain C         |
            +--------+------------------------------------+
                     +
                     +
                     +   Or-Or
                     +
            +--------+------------------------------------+
            |        +                                    |
            |        +                    +-------------+ |
            |        +             +++++++   VNFM2-1   | |
            |    +---+-------+     +      +-------------+ |
            |    |  NFVO-2   ++++++                       |
            |    +-----------+     +      +-------------+ |
            |                       +++++++   VNFM2-2   | |
            |                             +-------------+ |
            |             Administrative domain A         |
            +---------------------------------------------+
```

Figure 9: Architecture for management of composite and nested NS

5.2. Hierarchical

Considering the potential split of the NFVO into a Network Service Orchestrator (NSO) and a Resource Orchestrator (RO), multi-provider hierarchical interfaces may exist at their northbound APIs. Figure 10 illustrates the various interconnection options, namely:

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

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

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

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

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

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

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

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

```
                                             Tenants
                             *    Provider      |
      *                       *   Domain 4   +--+-----------+
       *                       *             |MP-NFVO/NSO:  |
        *                       *            |Network Serv. |
         *   Provider            *           |Orchestrator  |
          *  Domain 3             *          +--+-----------+
           *              Tenants  *            |E/RO
            *                |      ************|*************
             *              ++-------------+    |
              *             |MP-NFVO/NSO:  |    |
      Provider *            |Network Serv. |    |
      Domain 1  *           |Orchestrator  |    |
                 *          +-+-----+------+    |
                  *      E/NSO|     | I/RO     /
                   *.---------'   +-+---------+--+
                   /*             |MP-NFVO/RO:   |
                  /  *            |Resource      |
   Tenants       /    *           |Orchestrator  |
   |             |     *          +--+---+-------+
   | +-----------+--+   *************|***|********************
   | |MP-NFVO/NSO:  |                |  * \           Provider
   | |Network Serv. |         E/RO  /   *  \ E/VIM    Domain 2
   | |Orchestrator  |  .-----------'    *   '-------.
   | +-+------+-----+  |                *           |
   |   |I/NSO |I/RO    |                *           |
   |   |   +--+--------+--+             *           |
   |   |   |MP-NFVO/RO:   |             *           |
   |   |   |Resource      |             *           |
    \  |   |Orchestrator  |             *    +------+-------+
     \ |   +----+---- --+-+             *    |VIM:          |
    +--+-----+  |I/RO   |I/VIM          *    |Virtualized   |
    |NFVO/NSO|  |       |               *    |Pys mapping   |
    +------+-+  |       |               *    +--------------+
       I/RO|    |       |               *
    +------+----+---+   |               *
    |    NFVO/RO    |   |               *
    ++-------------++   |               *
     |Nfvo-Vim     |    |               *
    ++-------+    ++----+--+            *
    |WIM|VIM ||   |VIM|WIM |            *
    +--------+|   +--------+            *
     +--------+                         *
```

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

5.3. Cascading

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

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

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

Orchestration operation in multi-domain is somewhat different from that in a single domain as the assumption in single domain single provider orchestration is that the orchestrator is aware of the entire topology and resource availability within its domain as well as has complete control over those resources. This assumption of technical control cannot be made in a multi domain scenario, furthermore the assumption of the knowledge of the resources and topologies cannot be made across providers. In such a scenario solutions are required that enable the exchange of relevant information across these orchestrators. This exchange needs to be standardized as shown in Figure 11.

```
                   |                                |
                   + IF1                            +
              _____|____                        ____|_____
             |   Multi  |        IF2           |   Multi  |
             | Provider |<--------+---------->| Provider |
             |___Orch___|                      |___Orch___|
                  /\                                /\
                 /  \                              /  \
                /    \ IF3                        /    \
        _______/__   _\_________        ________/_    _\________
       |  Domain  | |  Domain   |      |  Domain  |  |  Domain  |
       |___Orch___| |___Orch___|       |___Orch___|  |___Orch___|
```

Figure 11: Multi Domain Multi Provider reference architecture

The figure shows the Multi Provider orchestrator exposing an interface 1 (IF1) to the tenant, interface 2 (IF2) to other Multi Provider Orchestrator (MPO) and an interface 3 (IF3) to individual domain orchestratrators. Each one of these interfaces could be a possible standardization candidate. Interface 1 is exposed to the tenant who could request his specific services and/or slices to be deployed. Interface 2 is between the orchestrator and is a key interface to enable multi-provider operation. Interface 3 focuses on abstracting the technology or vendor dependent implementation details to support orchestration.

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

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

```
                   |                                    |
                   + IF1                                +
     ______________|______________                  ____|_____
    |      Multi Provider Orch    |                |  Multi    |
    | ______   ________   _______ |<------+------->| Provider |
    ||Assur-| |        | | Catal-||      IF2       |___Orch___|
    ||-ance | |  NFVO  | | logue ||
    || Mgmt.| |        | | Topo. ||
    ||______| |________| |_Mgmt._||
    |_____________________________|
                  /\
                 /  \ IF3
```

Figure 12: Detailed MPO reference architecture

The catalogue and topology management system is responsible for step 1. It discovers the service as well as the resources exposed by the other domains both on IF2 and IF3. The combination of these services with coverage over the detected topology is provided to the user over IF1. In turn the catalogue and topology management system is also responsible for exposing the topology and service deployment capabilities to the other domain. The exposure over interface 2 to other MPO maybe abstracted and the mapping of this abstracted view to the real view when requested by the NFVO.

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

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

6.1. Interworking interfaces

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

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

6.2. 5GEx Multi Architecture

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

```
                         ^                          ^
                    I1-S |                          |
                    I1-F |                     I1-C |
                    I1-RM|                          |
   +----------------------------------------------------+
   |          +-------------------------------------|--+|
   |          |          |                          |  ||     I2-S
   |          | +--------------------+              |  ||     I2-F
   |+---+     | | +-----+ +---+ IP-  |              |  ||     I2-RC
   ||OSS|<----|-| | NSO | |RO | NFVO +<-------------|--+|-------------->
   |+---+     | | +-----+ +---+      |<-------------+  ||
   |  ^       | +---^----------------+              |  ||
   |  |       |     |     ^ ^    ^^ ^               |  ||
   |  |       | +---+---+ | |    || |               |  ||
   |  +---------| VNF    | | |    || |  Multi-       |  ||
   |          | |Manager| | |    || |  Provider     |  ||
   |          | ++-------+ | |    || |  Orchestrator |  ||
   |          |  ^         | |    || |  (MPO)        |  ||
   |          |  +--------+ |    || |               |  ||
   |          |  |     +-------+ || |               |  ||    I2-Mon
   |          |  |     |SLA    |<-|-|---------------|--+|-------------->
   |          |  |     |Manager| || |               |  ||
   |          |  |     +-------+ || |               |  ||
   |          |  |       ^       || +------------+  |  ||I2-RT-advertise
   |          |  |       |       || |Topology    |  |  ||I2-RT-bilateral
   |          |  |       |       || |Distribution|<-|--+|-------------->
   |          |  |       |       || |Repository  |  |  ||
   |          |  |       |       || +--------^+--+  |  ||
   |          |  |       |       ||   ^      ||     |  ||
   |          |  |       |       ||   |  +---+v-+   |  ||I2-RC-network
   |          |  |       |       |+---|--+MD-PCE|<--|--+|-------------->
   |          |  |       |       |    |  +------+   |  ||
```

```
   |          |  |       |       |    |   ^ +-------+-+||I2-C-advertise
   |          |  |       |       |    |   | |Service  |||I2-C-bilateral
   |          |  |       |       |    |   | |Catalogue+<|-------------->
   |          |  |       |       |    |   | +---------+||
   |          |  |       |       |    |   |      ^    ||
   |          +--|----- -|-------|----|---|------|-----+|
   |             |       |            |   |      |      |
   |             |I3-RC  |   I3-S|    |   |I3-RC-network|
   |          +--+--+    |   +----+    | +---+     |      |
   |          | VIM |    |   |NFVO|    | |PCE|     |      |
   |          +-----+    |   +----+    | +---+     |      |
   |                     |            |           |      |
   |                     |       I3-RT|           |I3-C  |
   |              I3-Mon |     +------+----+ +---+-----+|
   |           +---------+-+   |Topology   | |Service  ||
   |Operator   | Monitoring|   |Abstraction| |Catalogue||
   |Domain     +-----------+   +-----------+ +---------+|
   +----------------------------------------------------+
```

Figure 13: 5GEx MPO functional architecture

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

The Inter-provider NFVO, including the RO and the NSO, implementing the multi-provider service decomposition

the VNF/Element manager, managing VNF lifecycle, scaling and responsible for FCAPS (Fault, Configuration, Accounting, Performance and Security management)

the SLA Manager, in charge of reporting monitoring and performance alerts on the service graph

the Service Catalogue, exposing available services to external client and operators

the Topology and Resource Distribution module and Repository, exchanging operators topologies (both IT and network resources) and providing abstracted view of the own operator topology

the Multi-domain Path Computation Element (PCE implementing inter-operator path computation to allow QoS-based connectivity serving VNF-VNF link).

The Inter-provider NVFO selects providers to be involved in the service chained request, according to policy-based decisions and resorting to Inter-Provider topologies and service catalogues advertised through interfaces I2-RT-advertise and I2-C-advertise, respectively. Network/service requests are sent to other providers using the I2-RC and I2-S interfaces, respectively. Policy enforcement for authorized providers running resource orchestration and lifecycle management are exploited through interfaces I2-RC and I2-F, respectively. The VNF/Element Manager is in charge of managing the lifecycle of the VNFs part of the services. More specifically, it is in charge to perform: the configuration of the VNFs, also in terms of security aspects, the fault recovery and the scaling according to their performance. The SLA Manager collects and aggregates quality measurement reports from probes deployed by the Inter-Provider NFVO as part of the service setup. Measurements results at the Manager represent aggregated results and are computed and stored utilizing the I2-Mon interface between Inter-Provider MPOs sharing the same service. Faults and alarms are moreover correlated to raise SLA violation to remote inter-provider MPOs and, optionally, to detect the source and the location of the violation, triggering service re-computation/rerouting procedures. The Service Catalogue stores information on network services and available VNFs and uses I2-C interfaces (either bilateral or advertised) to advertise and updating such offered services to other operators. To enable inter-provider service decomposition, multi-operator topology and peering relationships need to be advertised. Providers advertise basic inter-provider topologies using the I2-RT-advertse interface including, optionally, abstracted network resources, overall IT resource capabilities, MPO entry-point and MD-PCE IP address. Basic advertisement takes place between adjacent operators. These information are collected, filtered by policy rules and propagated hop-by-hop. In 5GEx, the I2-RT-advertise interfaces utilizes BGP-LS protocol. Moreover, providers establish point-to-point bilateral (i.e., direct and exclusive) communications to exchange additional topology and business information, using the I2-RT-bilateral interface. Service decomposition may imply the instantiation of traffic-engineered multi-provider connectivity, subject to constraints such as guaranteed bandwidth, latency or minimum TE metric. The multi-domain PCE (MD-PCE) receives the connectivity request from the inter-provider NFVO and performs inter-operator path computation to instantiate QoS-based connectivity between two VNFs (e.g., Label Switched Paths). Two procedures are run sequentially:

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

per-operator connectivity computation and instantiation.

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

6.3. 5G-TRANSFORMER Architecture

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

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

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

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

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

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

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

```
            _________                        _________
           |         |                      |         |
           |   VS    |                      |   VS    |
           |_________|                      |_________|
                |                                |
                + IF1                            +
            ____|____                        ____|____
           |         |         IF2          |         |
           |   SO    |<--------+----------->|   SO    |
           |_________|                      |_________|
               /\                               /\
              /  \                             /  \
             /    \ IF3                       /    \
       ______/__  _\_______              _______/_   _\_______
      |   MTP   | |   MTP   |           |   MTP   | |   MTP   |
      |_________| |_________|           |_________| |_________|
```

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

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

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

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

6.3.1. So-Mtp Interface (IF3)

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

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

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

So-Mtp(-RMM). It provides the required primitives and parameters for supporting the SO resource monitoring management (RMM) capability for the purpose of fault management and SLA assurance handled by assurance management component in Figure 14.

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

6.3.2. So-So Interface (IF2)

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

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

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

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

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

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

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

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

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

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

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

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

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

6.3.3. Vs-So Interface (IF1)

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

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

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

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

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

7. Multi-domain orchestration and Open Source

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

At this moment several open source industry projects were formed to develop integrated NFV orchestration platform. The most known of them are ONAP [onap], OSM [osm] and Cloudify [cloudify]. While all these projects have different drivers, motivations, implementation approach and technology stack under the hood, all of them are considering multi-VIM deployment scenario, i.e. all these software platforms are capable to deploy NFV service over different virtualized infrastructures, like public or private providers. Additionally OSM and Cloudify orchestration platforms have capabilities to manage interconnection among managed VIMs using appropriate plugins or drivers. However, despite the fact that typical Telco/Carrier infrastructure has multiple domains (both technology and administrative), none of these orchestration projects is focused on a service federation use case development.

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

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

8. IANA Considerations

N/A.

9. Security Considerations

TBD.

10. Acknowledgments

This work is supported by 5G-PPP 5GEx, an innovation action project partially funded by the European Community under the H2020 Program (grant agreement no. 671636). This work is also supported by 5G-PPP 5G-TRANSFORMER, a research and innovation action project partially funded by the European Community under the H2020 Program (grant agreement no. 761536). The views expressed here are those of the

authors only. The European Commission is not liable for any use that may be made of the information in this presentation.

11. Informative References

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

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

Email: luismiguel.conterasmurillo@telefonica.com

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

Email: Ishan.vaishnavi@huawei.com

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

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

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

Email: josep.mangues@cttc.cat

Xi Li
NEC
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Email: Xi.Li@neclab.eu

Francesco Paolucci
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

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

Andrea Sgambelluri
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

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

Barbara Martini
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Email: barbara.martini@cnit.it

Luca Valcarenghi
SSSA
Via Giuseppe Moruzzi, 1
Pisa 56121
Italy

Email: luca.valcarenghi@santannapisa.it

Giada Landi
Nextworks
Via Livornese, 1027
Pisa 56122
Italy

Email: g.landi@nextworks.it

Dmitriy Andrushko
MIRANTIS

Email: dandrushko@mirantis.com

Alain Mourad
InterDigital Europe

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

NFV RG CJ. Bernardos
Internet-Draft UC3M
Intended status: Experimental A. Mourad
Expires: September 6, 2018 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 will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

The telecommunications sector is experiencing a major revolution that will shape the way networks and services are designed and deployed for the next decade. We are witnessing an explosion in the number of applications and services demanded by users, which are now really capable of accessing them on the move. In order to cope with such a demand, some network operators are looking at the cloud computing paradigm, which enables a potential reduction of the overall costs by outsourcing communication services from specific hardware in the operator's core to server farms scattered in 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.

```
+--------------------------------------------+  +----------------+
|   Virtualized Network Functions (VNFs)     |  |                |
|  -------   -------   -------   -------     |  |                |
|  |     |   |     |   |     |   |     |     |  |                |
|  | VNF |   | VNF |   | VNF |   | VNF |     |  |                |
|  |     |   |     |   |     |   |     |     |  |                |
|  -------   -------   -------   -------     |  |                |
+--------------------------------------------+  |                |
                                                |                |
+--------------------------------------------+  |                |
|         NFV Infrastructure (NFVI)          |  |      NFV       |
| -----------    -----------    ----------- |  |   Management   |
| | Virtual |    | Virtual |    | Virtual | |  |      and       |
| | Compute |    | Storage |    | Network | |  |  Orchestration |
| -----------    -----------    ----------- |  |                |
| +----------------------------------------+ |  |                |
| |         Virtualization Layer           | |  |                |
| +----------------------------------------+ |  |                |
| +----------------------------------------+ |  |                |
| | -----------  -----------  ----------- | |  |                |
| | | Compute |  | Storage |  | Network | | |  |                |
| | -----------  -----------  ----------- | |  |                |
| |           Hardware resources           | |  |                |
| +----------------------------------------+ |  |                |
+--------------------------------------------+  +----------------+
```

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).

```
                                                  +--------------------+
   +-------------------------------------------+  | ----------------   |
   |                 OSS/BSS                   |  | | NFV          |   |
   +-------------------------------------------+  | | Orchestrator +-- |
                                                  | ---+------------ | |
   +-------------------------------------------+  |    |             | |
   |  ---------     ---------     ---------    |  |    |             | |
   |  | EM 1  |     | EM 2  |     | EM 3  |    |  |    |             | |
   |  ----+----     ----+----     ----+----    |  | ---+----------   | |
   |      |             |             |        |--|-|    VNF     |   | |
   |  ----+----     ----+----     ----+----    |  | | manager(s) |   | |
   |  | VNF 1 |     | VNF 2 |     | VNF 3 |    |  | ---+----------   | |
   |  ----+----     ----+----     ----+----    |  |    |             | |
   +------|-------------|-------------|--------+  |    |             | |
          |             |             |           |    |             | |
   +------+-------------+-------------+--------+  |    |             | |
   |         NFV Infrastructure (NFVI)         |  |    |             | |
   | -----------    -----------    ----------- |  |    |             | |
   | | Virtual |    | Virtual |    | Virtual | |  |    |             | |
   | | Compute |    | Storage |    | Network | |  |    |             | |
   | -----------    -----------    ----------- |  | ---+------       | |
   | +---------------------------------------+ |  | |        |       | |
   | |         Virtualization Layer          | |--|-| VIM(s) +-------- |
   | +---------------------------------------+ |  | |        |         |
   | +---------------------------------------+ |  | ----------         |
   | | -----------  -----------  ----------- | |  |                    |
   | | | Compute |  | Storage |  | Network | | |  |                    |
   | | | hardware|  | hardware|  | hardware| | |  |                    |
   | | -----------  -----------  ----------- | |  |                    |
   | |          Hardware resources           | |  |  NFV Management    |
   | +---------------------------------------+ |  | and Orchestration  |
   +-------------------------------------------+  +--------------------+
```

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 ir 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 resouces (as described in [I-D.aranda-sfc-dp-mobile]).

```
            --------                        ---------   ---------
   control  | ctr1 |........................| ctrl2 |...| ctrl3 |
   plane    --------                        ---------   ---------
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                ------         ------     ------
                               .| f3 |.........| f5 |.....| f6 |
          ------       ------ . ------         ------     ------
    user  | f1 |.......| f2 |.                    .
   plane  ------       ------ . ------            .
                               .| f4 |.............
                                ------
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         +--------------------------------+  +--------------------+
         | -------   --------   --------  |  |      ----------    |
         | |     |   |      |   |      |  |  |    ---------- |    |
         | | @UE |   | @car |   | @eNB |  |  |  ---------- | |    |
         | -------   --------   --------  |  |  |  Data  | | |    |
         |                                |  |  | Center | | -    |
         | -------- Heterogeneous ------- |  |  |  (DC)  |-       |
    phy  | |      |   computing   |     | |  |  ----------        |
   infra | |@train|    devices    | @AP | |==|      ----------    |
         | --------    forming    ------- |  |    ---------- |    |
         |             the fog            |  |  ---------- | |    |
         | ---------        ------------  |  |  |  Data  | | |    |
         | |       |        |          |  |  |  | Center | | -    |
         | | @mall |        | @localDC |  |  |  |  (DC)  |-       |
         | ---------        ------------  |  |  ----------        |
         |              FOG               |  |       CLOUD        |
         +--------------------------------+  +--------------------+
         <--------- fog and edge ----------------->
                                     <--- edge & central cloud --->
```

Figure 3: Fog virtualization

5. Problem statemement

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.

```
                                                             __
                               ___________                 _(  )_
    ------------  _(           )_    -----------  _(        )_
    | terminal |-(_  VIM--NFVI  _)   | network |-(_  NFVO  _)
    ------------   (___________)     -----------   (_     _)
        |                                |           (__)
       XXX (1. attachment)               |
        |                                |
        +---2. Advertisement----------->|
        |                                |
        |<......(3. VIM Registration)...+
        |                                |
```

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.

```
                                                      ___________
                                                    _(           )_
                                                  _(      +-NFVI   )_
    ------------  _(      )_       -----------  _(       /           )_
    | terminal |-(_  NFVO  _)      | network |-(_  VIM(s)---NFVI      _)
    ------------   (______)        -----------   (_      \          _)
        |                               |          (_     +-NFVI  _)
       XXX (1. attachment)              |            (___________)
        |                               |
        +---2. Request----------------->|
        |                               |
        |<-----------3. Advertisement---|
        |                               |
        +...(4. VIM Registration)......>|
        |                               |
```

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 CJ. Bernardos
Internet-Draft UC3M
Intended status: Informational A. Rahman
Expires: March 6, 2019 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

Table of Contents

1. Introduction and scope

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

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

This document presents research and engineering challenges in network virtualization that need to be addressed in order to achieve these goals, spanning from pure research and engineering/standards space. The objective of this memo is to document the technical challenges and corresponding current approaches and to expose requirements that should be addressed by future research and standards work.

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

2. Terminology

The following terms used in this document are defined by the ETSI Network Function Virtualization (NVF) Industrial Study Group (ISG) [etsi_gs_nfv_003], the ONF [onf_tr_521] and the IETF [RFC7426] [RFC7665]:

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

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

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

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

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

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

NFV Orchestrator (NFVO): functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.

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

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

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

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

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

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

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

3. Background

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

3.1. Network Function Virtualization

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

- 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.

```
   +-------------------------------------------+  +---------------+
   |   Virtualized Network Functions (VNFs)    |  |               |
   |  -------   -------   -------   -------    |  |               |
   |  |     |   |     |   |     |   |     |    |  |               |
   |  | VNF |   | VNF |   | VNF |   | VNF |    |  |               |
   |  |     |   |     |   |     |   |     |    |  |               |
   |  -------   -------   -------   -------    |  |               |
   +-------------------------------------------+  |               |
                                                  |               |
   +-------------------------------------------+  |               |
   |         NFV Infrastructure (NFVI)         |  |      NFV      |
   | -----------    -----------    ----------- |  |  Management   |
   | | Virtual |    | Virtual |    | Virtual | |  |      and      |
   | | Compute |    | Storage |    | Network | |  | Orchestration |
   | -----------    -----------    ----------- |  |               |
   | +---------------------------------------+ |  |               |
   | |         Virtualization Layer          | |  |               |
   | +---------------------------------------+ |  |               |
   | +---------------------------------------+ |  |               |
   | | -----------  -----------  ----------- | |  |               |
   | | | Compute |  | Storage |  | Network | | |  |               |
   | | -----------  -----------  ----------- | |  |               |
   | |          Hardware resources           | |  |               |
   | +---------------------------------------+ |  |               |
   +-------------------------------------------+  +---------------+
```

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).

```
                                                  +--------------------+
   +-------------------------------------------+  | ----------------   |
   |                 OSS/BSS                   |  | | NFV          |   |
   +-------------------------------------------+  | | Orchestrator +-- |
                                                  | ---+------------ | |
   +-------------------------------------------+  |    |             | |
   |  ---------     ---------     ---------    |  |    |             | |
   |  | EM 1  |     | EM 2  |     | EM 3  |    |  |    |             | |
   |  ----+----     ----+----     ----+----    |  | ---+----------   | |
   |      |             |             |        |--|-|    VNF     |   | |
   |  ----+----     ----+----     ----+----    |  | | manager(s) |   | |
   |  | VNF 1 |     | VNF 2 |     | VNF 3 |    |  | ---+----------   | |
   |  ----+----     ----+----     ----+----    |  |    |             | |
   +------|-------------|-------------|--------+  |    |             | |
          |             |             |           |    |             | |
   +------+-------------+-------------+--------+  |    |             | |
   |         NFV Infrastructure (NFVI)         |  |    |             | |
   | -----------    -----------    ----------- |  |    |             | |
   | | Virtual |    | Virtual |    | Virtual | |  |    |             | |
   | | Compute |    | Storage |    | Network | |  |    |             | |
   | -----------    -----------    ----------- |  | ---+------       | |
   | +---------------------------------------+ |  | |        |       | |
   | |         Virtualization Layer          | |--|-| VIM(s) +-------- |
   | +---------------------------------------+ |  | |        |         |
   | +---------------------------------------+ |  | ----------         |
   | | -----------  -----------  ----------- | |  |                    |
   | | | Compute |  | Storage |  | Network | | |  |                    |
   | | | hardware|  | hardware|  | hardware| | |  |                    |
   | | -----------  -----------  ----------- | |  |                    |
   | |          Hardware resources           | |  |  NFV Management    |
   | +---------------------------------------+ |  | and Orchestration  |
   +-------------------------------------------+  +--------------------+
```

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.

```
+----------+
| -------  |
| |Oper.|  |           O
| |Mgmt.|  |<........> -+- Network Operator
| |Iface|  |           ^
| -------  |      +----------------------------------------+
|          |      | +------------------------------------+ |
|          |      | | ---------  ---------     --------- | |
|--------- |      | | | App 1 |  | App 2 | ... | App n | | |
||Plugins| |<....>| | ---------  ---------     --------- | |
|--------- |      | | Plugins                            | |
|          |      | +------------------------------------+ |
|          |      | Application Plane                      |
|          |      +----------------------------------------+
|          |                        A
|          |                        |
|          |                        V
|          |      +----------------------------------------+
|          |      | +------------------------------------+ |
|--------- |      | |     ------------  ------------     | |
|| Netw. | |      | |     | Module 1 |  | Module 2 |     | |
||Engine | |<....>| |     ------------  ------------     | |
|--------- |      | | Network Engine                     | |
|          |      | +------------------------------------+ |
|          |      | Controller Plane                       |
|          |      +----------------------------------------+
|          |                        A
|          |                        |
|          |                        V
|          |      +----------------------------------------+
|          |      |  +---------------+   +---------------+ |
|          |      |  | ------------  |   | ------------  | |
|----------|      |  | | OpenFlow |  |   | | OpenFlow |  | |
||OpenFlow||<....>|  | ------------  |   | ------------  | |
|----------|      |  | NE            |   | NE            | |
|          |      |  +---------------+   +---------------+ |
|          |      | Data Plane                             |
|Management|      +----------------------------------------+
+----------+
```

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).

```
                  o--------------------------------o
                  |                                |
                  | +-------------+   +----------+ |
                  | | Application |   |  Service | |
                  | +-------------+   +----------+ |
                  |       Application Plane        |
                  o---------------Y----------------o
                                  |
    *-----------------------------Y---------------------------------*
    |           Network Services Abstraction Layer (NSAL)           |
    *------Y------------------------------------------------Y-------*
           |                                                |
           |               Service Interface                |
           |                                                |
    o------Y------------------o       o---------------------Y------o
    |      |    Control Plane |       | Management Plane    |      |
    | +----Y----+   +-----+   |       |  +-----+       +----Y----+ |
    | | Service |   | App |   |       |  | App |       | Service | |
    | +----Y----+   +--Y--+   |       |  +--Y--+       +----Y----+ |
    |      |           |      |       |     |               |      |
    | *----Y-----------Y----* |       | *---Y---------------Y----* |
    | | Control Abstraction | |       | | Management Abstraction | |
    | |      Layer (CAL)    | |       | |      Layer (MAL)       | |
    | *----------Y----------* |       | *----------Y-------------* |
    |            |            |       |            |               |
    o------------|------------o       o------------|---------------o
                 |                                 |
                 | CP                              | MP
                 | Southbound                      | Southbound
                 | Interface                       | Interface
                 |                                 |
    *------------Y---------------------------------Y----------------*
    |         Device and resource Abstraction Layer (DAL)           |
    *------------Y---------------------------------Y----------------*
    |            |                                 |                |
    |    o-------Y----------o   +-----+   o--------Y----------o     |
    |    | Forwarding Plane |   | App |   | Operational Plane |     |
    |    o------------------o   +-----+   o-------------------o     |
    |                        Network Device                         |
    +---------------------------------------------------------------+
```

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.

```
          MMFO                          MMFA
   +-----+ . +---------------------+ . +--------------------+
   |     | . |+---+ +---+ +-------+| . |+---------+ +-----+ |
   |     | . ||   | |   | |       || . ||   AL.   | |     | |
   |     | . || E | |   | |  App. || . || mngmt.  | | SDN | | SDN-AL
   |     | . || x | | M | | layer || . || support | | app | |
   |     | . || t.| | u | | Mngmt.|| . || & orch. | |     | |
   |     | . ||   | | l | +-------+| . |+---------+ +-----+ |
   |     | . || r | | t |          | . +--------------------+
   |     | . || e | | i |          |MMFC ..................... ACI
   |     | . || l | |   |          | . +--------------------+
   |     | . || a | | l | +-------+| . |+------+ +---------+|
   | OSS/| . || t | | a | |       || . ||      | |   App.  ||
   | BSS | . || i | | y | |       || . ||      | | support ||
   |     | . || o | | e | |       || . ||      | +---------+|
   |     | . || n | | r | |       || . ||  CL  | +---------+|
   |     | . || s | |   | |Control|| . ||mngmt.| | Control ||
   |     | . || h | | m | | layer || . || supp.| |  layer  || SDN-CL
   |     | . || i | | a | | mngmt.|| . || and  | |  serv.  ||
   |     | . || p | | n | |       || . || orch.| +---------+|
   |     | . ||   | | a | |       || . ||      | +---------+|
   |     | . || m | | g | |       || . ||      | | Resource||
   |     | . || n | | e | |       || . ||      | | abstrac.||
   |     | . || g | | m | +-------+| . |+------+ +---------+|
   |     | . || m | | e |          | . +--------------------+
   |     | . || t.| | n |          |MMFR ..................... RCI
   |     | . ||   | | t |          | . +--------------------+
   +-----+ . |+---+ |   | +-------+| . |+------++----------+|
             |      | o | |       || . ||      ||RL control||
             |      | r | |Resour.|| . ||  RL  |+----------+|
        MMF  |      | c | | layer || . ||mngmt.|+----++----+| SDN-RL
             |      | h.| | mngmt.|| . || supp.||Data||Data||
             |      |   | |       || . ||      ||tran||proc||
             |      +---+ +-------+| . |+------++----++----+|
             +---------------------+ . +--------------------+
```

Figure 5: ITU-T SDN functional architecture

3.4. Multi-access Edge Computing

Multi-access Edge Computing (MEC) -- formerly known as Mobile Edge Computing -- capabilities deployed in the edge of the mobile network can facilitate the efficient and dynamic provision of services to mobile users. The ETSI ISG MEC working group, operative from end of 2014, intends to specify an open environment for integrating MEC capabilities with service providers' networks, including also applications from 3rd parties. These distributed computing capabilities will make available IT infrastructure as in a cloud environment for the deployment of functions in mobile access networks. It can be seen then as a complement to both NFV and SDN.

3.5. IEEE 802.1CF (OmniRAN)

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

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

3.6. Distributed Management Task Force

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

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

3.7. Open Source initiatives

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

- OpenStack. OpenStack is a free and open-source cloud-computing software platform. OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API.

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 Telefonica’s OpenMANO or Canonical’s Juju, among others.

- OpenBaton. OpenBaton is a ETSI NFV compliant Network Function Virtualization Orchestrator (NFVO). OpenBaton was part of the OpenSDNCore project started with the objective of providing a compliant implementation of the ETSI NFV specification.

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

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

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

4. Network Virtualization Challenges

4.1. Introduction

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

4.2. Guaranteeing quality-of-service

Achieving a given quality-of-service in an NFV environment with virtualized and distributed computing, storage and networking functions is more challenging than providing the equivalent in discrete non-virtualized components. For example, ensuring a guaranteed and stable forwarding data rate has proven not to be straightforward when the forwarding function is virtualized and runs on top of COTS server hardware [openmano_dataplane] [I-D.mlk-nfvrg-nfv-reliability-using-cots] [etsi_nvf_whitepaper_3]. Again, the comparison point is against a router or forwarder built on optimized hardware. We next identify some of the challenges that this poses.

4.2.1. Virtualization Technologies

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

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

4.2.2. Metrics for NFV characterization

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

The IP Performance Metrics (IPPM) WG defines metrics that can be used to measure the quality and performance of Internet services and applications running over transport layer protocols (e.g., TCP, UDP) over IP. It also develops and maintains protocols for the measurement of these metrics. While the IPPM WG is a long running WG that started in 1997, at the time of writing it does not have a charter item or active drafts related to the topic of network virtualization. In addition to using IPPM metrics to evaluate the QoS, there is a need for specific metrics for assessing the performance of network virtualization techniques.

The Benchmarking Methodology Working Group (BMWG) is also performing work related to NFV metrics. For example, [RFC8172] investigates additional methodological considerations necessary when benchmarking VNFs instantiated and hosted in general-purpose hardware, using bare-metal hypervisors or other isolation environments such as Linux containers. An essential consideration is benchmarking physical and virtual network functions in the same way when possible, thereby allowing direct comparison.

As stated in the document [RFC8172], there is a clear motivation for the work on performance metrics for NFV [etsi_gs_nfv_per_001], that is worth replicating here: "I’m designing and building my NFV Infrastructure platform. The first steps were easy because I had a small number of categories of VNFs to support and the VNF vendor gave HW recommendations that I followed. Now I need to deploy more VNFs from new vendors, and there are different hardware recommendations. How well will the new VNFs perform on my existing hardware? Which among several new VNFs in a given category are most efficient in terms of capacity they deliver? And, when I operate multiple categories of VNFs (and PNFs) *concurrently* on a hardware platform such that they share resources, what are the new performance limits, and what are the software design choices I can make to optimize my chosen hardware platform? Conversely, what hardware platform upgrades should I pursue to increase the capacity of these concurrently operating VNFs?"

Lately, there are also some efforts looking into VNF benchmarking. The selection of an NFV Infrastructure Point of Presence to host a VNF or allocation of resources (e.g., virtual CPUs, memory) needs to be done over virtualized (abstracted and simplified) resource views [vnf_benchmarking] [I-D.rorosz-nfvrg-vbaas].

4.2.3. Predictive analysis

On top of diagnostic tools that enable an assessment of the QoS, predictive analyses are required to react before anomalies occur. Due to the SW characteristics of VNFs, a reliable diagnosis framework could potentially enable the prevention of issues by a proper diagnosis and then a reaction in terms of acting on the potentially impacted service (e.g., migration to a different compute node, scaling in/out, up/down, etc).

4.2.4. Portability

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

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

4.3. Performance improvement

4.3.1. Energy Efficiency

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

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

- Energy aware scaling (e.g., reductions in CPU speeds and partially turning off some hardware com- ponents to meet a given energy consumption target.

- Energy-aware function placement.

- Scheduling and chaining algorithms, for example adapting the network topology and operating parameters to minimize the operation cost (e.g., tracking energy costs to identify the cheapest prices).

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

4.3.2. Improved link usage

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

4.4. Multiple Domains

Market fragmentation has resulted in a multitude of network operators each focused on different countries and regions. This makes it difficult to create infrastructure services spanning multiple countries, such as virtual connectivity or compute resources, as no single operator has a footprint everywhere. Cross-domain orchestration of services over multiple administrations or over multi-domain single administrations will allow end-to-end network and service elements to mix in multi-vendor, heterogeneous technology and resource environments [multi-domain_5GEx].

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

Multi-domain orchestration is currently an active research topic, which is being tackled, among others, by ETSI NFV ISG and the 5GEx project (https://www.5gex.eu/) [I-D.bernardos-nfvrg-multidomain] [multi-domain_5GEx].

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

4.5. 5G and Network Slicing

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

Because of these very extreme 5G use cases, it is envisioned that selective combinations of radio access networks and core network components will have to be combined into a given network slice to address the specific requirements of each use case.

For example, within the major IoT category, which is perhaps the most disrupting one, some autonomous IoT devices will have very low throughput, will have much longer sleep cycles (and therefore high latency), and a battery life time exceeding by a factor of thousands that of smart phones or some other devices that will have almost continuous control and data communications. Hence, it is envisioned that a customized network slice will have to be stitched together from virtual resources or sub-slices to meet these requirements.

The actual definition of network slice from an IP infrastructure viewpoint is currently undergoing intense debate [I-D.geng-coms-problem-statement] [I-D.gdmb-netslices-intro-and-ps] [I-D.defoy-netslices-3gpp-network-slicing] [ngmn_5G_whitepaper]. Network slicing is a key for introducing new actors in existing market at low cost -- by letting new players rent "blocks" of capacity, if the new business model enables performance that meets the application needs (e.g., broadcasting updates to many sensors with satellite broadcasting capabilities). However, more work needs to be done to define the basic architectural approach of how network slices will be defined and formed. For example, is it mostly a matter of defining the appropriate network models (e.g. YANG) to stitch the network slice from existing components. Or do end-to-end timing, synchronization and other low level requirements mean that more fundamental research has to be done.

4.5.1. Virtual Network Operators

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

VNOs aim to exploit the virtualized infrastructures to deliver new and improved services to their customers. However, current network virtualization techniques offer poor support for VNOs to control their resources. It has been considered that the InP is responsible for the reliability of the virtual resources but there are several situations in which a VNO requires to gain a finer control on its resources. For instance, dynamic events, such as the identification of new requirements or the detection of incidents within the virtual system, might urge a VNO to quickly reform its virtual infrastructure and resource allocation. However, the interfaces offered by current virtualization platforms do not offer the necessary functions for VNOs to perform the elastic adaptations they require to tackle with their dynamic operation environments.

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

Such common methods should be offered by all underlying controllers, regardless of being network-oriented (e.g. ODL, ONOS, Ryu) or computing-oriented (e.g. OpenStack, OpenNebula, Eucalyptus). Furthermore, it is also important for those platforms to offer some "PUSH" function to report resource state, avoiding the need for the VNO’s controller to "POLL" for such data. A starting point to get proper notifications within current REST APIs could be to consider the protocol proposed by the WEBPUSH WG [RFC8030].

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

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

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

Elsewhere, for 5G cellular evolution there is much discussion on the need for supporting virtual "network slices" for the expected massive numbers of IoT devices. A separate virtual network slice is considered necessary for different 5G IoT use cases because devices will have very different characteristics than typical cellular devices like smart phones [ngmn_5G_whitepaper], and the number of IoT devices is expected to be at least one or two orders of magnitude higher than other 5G devices (see Section 4.5).

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

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

4.6. Service Composition

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

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

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

As described in [dynamic_chaining], different algorithms can be used to enable dynamic service composition that optimizes a QoS-based utility function (e.g., minimizing the latency per-application traffic flows) for a given composition plan. Such algorithms can consider the computation capabilities and load status of resources executing the VNF instances, either deduced through estimations from historical usage data or collected through real-time monitoring (i.e., context-aware selection). For this reason, selections should include references to dynamic information on the status of the service instance and its constituent elements, i.e., monitoring information related to individual VNF instances and links connecting them as well as derived monitoring information at the chain level (e.g., end-to-end delay). At runtime, if one or more VNF instances are no more available or QoS degrades below a given threshold, the service selection task can be rerun to perform service substitution.

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

The SFC working group is working on an architecture for service function chaining [RFC7665] that includes the necessary protocols or protocol extensions to convey the Service Function Chain and Service Function Path information to nodes that are involved in the implementation of service functions and Service Function Chains, as well as mechanisms for steering traffic through service functions.

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

4.7. End-user device virtualization

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

4.8. Security and Privacy

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

In the case of security, there are situations where multiple service providers will need to coexist in a virtual or hybrid physical/virtual environment. This requires attestation procedures amongst different virtual/physical functions and resources, as well as ongoing external monitoring. Similarly, different network slices operating on the same infrastructure can present security problems, for instance if one slice running critical applications (e.g. support for a safety system) is affected by another slice running a less critical application. In general, the minimum common denominator for security measures on a shared system should be equal or higher than the one required by the most critical application. Multiple and continuous threat model analysis, as well as DevOps model are required to maintain a certain level of security in an NFV system. Simplistically, DevOps is a process that combines multiple functions into single cohesive teams in order to quickly produce quality software. It typically relies on also applying the Agile development process, which focuses on (among many things) dividing large features into multiple, smaller deliveries. One part of this is to immediately test the new smaller features in order to get immediate feedback on errors so that if present, they can be immediately fixed and redeployed.

On the other hand, privacy refers to concerns about the control of personal data and the decision of what to reveal to whom. In this case, the storage, transmission, collection, and potential correlation of information in the NFV system, for purposes not originally intended or not known by the user, should be avoided. This is particularly challenging, as future intentions and threats cannot be easily predicted, and still can be applied on data collected in the past. Therefore, well-known techniques such as data minimization, using privacy features as default, and allowing users to opt in/out should be used to prevent potential privacy issues.

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

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

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

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

4.9. Separation of control concerns

NFV environments offer two possible levels of SDN control. One level is the need for controlling the NFVI to provide connectivity end-to-end among VNFs or among VNFs and PNFs (Physical Network Functions). A second level is the control and configuration of the VNFs themselves (in other words, the configuration of the network service implemented by those VNFs), taking advantage of the programmability brought by SDN. Both control concerns are separated in nature.

However, interaction between both could be expected in order to optimize, scale or influence each other.

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

4.10. Network Function placement

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

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

4.11. Testing

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

1. Changes in methodology.

2. New functionality.

3. Opportunities.

4.11.1. Changes in methodology

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

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

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

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

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

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

- 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-tenenacy is a reality in virtualization, having the test system share resources with the SUT will impact the measured results in a performance test bed. The test system should be deployed on a separate platform in order to not to impact the resources available to the SUT.

4.11.2. New functionality

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

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

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

4.11.3. Opportunities

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

This new concept helps to enable modern concepts like DevOps and Continuous Integration and Continuous Deployment in the NFV environment. The CI/CD pipeline supports this concept. It consists of a series of tools, among which immediate testing is an integral part, to deliver software from source to deployment. The ability to deploy the test tools themselves into the production environment stretches the CI/CD pipeline all the way to production deployment, allowing a range of tests to be executed. The tests can be simple, with a goal of verifying the correct deployment and networking establishment, but can also be more complex, like testing VNF functionality.

5. Technology Gaps and Potential IETF Efforts

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

Open Research Area	Potential IETF/IRTF Group
1-Guaranteeing QoS	IPPM WG (Measurements of NFVI)
2-Performance improvement	SFC WG, NFVRG (energy driven orchestration)
3-Multiple Domains	NFVRG (multi-domain orchestration)
4-Network Slicing	NVO3 WG, NETSLICES bar BoF (multi-tenancy support)
5-Service Composition	SFC WG (SFC Mgmt and Config)
6-End-user device virtualization	N/A
7-Security	N/A
8-Separation of control concerns	NFVRG (separation between transport control and services)
9-Testing	NFVRG (testing of scaling)
10-Function placement	NFVRG, SFC WG (VNF placement algorithms and protocols)

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

6. NFVRG focus areas

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

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

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

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

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

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

Table 2: Mapping of NFVRG Focus Points to Open Research Areas

7. IANA Considerations

N/A.

8. Security Considerations

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

9. Acknowledgments

The authors want to thank Dirk von Hugo, Rafa Marin, Diego Lopez, Ramki Krishnan, Kostas Pentikousis, Rana Pratap Sircar, Alfred Morton, Nicolas Kuhn, Saumya Dikshit, Fabio Giust, Evangelos Haleplidis, Angeles Vazquez-Castro, Barbara Martini, Jose Saldana and Gino Carrozzo for their very useful reviews and comments to the document. Special thanks to Pedro Martinez-Julia, who provided text for the network slicing section.

The authors want to also thank Dave Oran and Michael Welzl for their very detailed IRSG reviews.

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

10. Informative References

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

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

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

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

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

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

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

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

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

[I-D.bernardos-nfvrg-multidomain]
Bernardos, C., Contreras, L., Vaishnavi, I., Szabo, R., Li, X., Paolucci, F., Sgambelluri, A., Martini, B., Valcarenghi, 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
Labege 31670
France

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

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

Email: luismiguel.contrerasmurillo@telefonica.com

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

Email: pedroandres.aranda@uc3m.es

Pierre Lynch
Ixia

Email: plynch@ixiacom.com

NMRG P. Martinez-Julia, Ed.
Internet-Draft NICT
Updates: draft-pedro-nmrg-anticipated- June 28, 2018
adaptation-01 (if approved)
Intended status: Informational
Expires: December 30, 2018

Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems
draft-pedro-nmrg-anticipated-adaptation-02

Abstract

The adoption of SDN/NFV technologies by current computer and network system infrastructures is constantly increasing, becoming essential for the the particular case of edge/branch network systems. The systems supported by these infrastructures require to be adapted to environment changes within a short period of time. Thus, the complexity of new systems and the speed at which management and control operations must be performed go beyond human limits. Thus, management systems must be automated. However, in several situations current automation techniques are not enough to respond to requirement changes. Here we propose to anticipate changes in the operation environments of SDN/NFV systems in response to external events and reflect it in the anticipation of the amount of resources required by those systems for their ulterior adaptaion. The final objective is to avoid service degradation or disruption while keeping close-to-optimum resource allocation to reduce monetary and operative cost as much as possible. Here we discuss how to achieve such capabilities by the integration of the Autonomic Resource Control Architecture (ARCA) to the management and operation (MANO) of NFV systems. We showcase it by building a multi-domain SDN/NFV infrastructure based on OpenStack and deploying ARCA to adapt a virtual system based on the edge/branch network concept to the operational conditions of an emergency support service, which is rarely used but that cannot leave any user unattended.

Status of This Memo

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

This Internet-Draft will expire on December 30, 2018.

Copyright Notice

Table of Contents

1. Introduction

The incorporation of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to current infrastructures to build virtual computer and network systems is constantly increasing. The need to automate the management and control of such systems has motivated us to design the Autonomic Resource Control Architecture (ARCA), as presented in ICIN 2018 [ICIN-2018]. Automation requirements are enough justified by the increasing size and complexity of systems, which in turn are essential in the current digital world. Moreover, the particular requirements and market benefits of network virtualization have been crystallized in the uprising of SDN/NFV infrastructures. Nowadays they broad reception of the combined SDN/NFV technology supposes a huge leap towards the empowerment and homogenization of virtualization technologies. Therefore, we have modeled ARCA to fit within the reference architecture for management and orchestration of NFV elements, the Virtual Network Functions (VNFs).

Behind the scenes, NFV is based on a highly distributed and network empowered version of the well-known Cloud infrastructures and platforms, also complemented by their centralized counterparts. This takes to virtual networks the high degree of flexibility already found for computer systems. It is highly desirable at the time NFV is being exploited by many organizations to build their private infrastructures, as well as by network service providers to build the services they later commercialize. However, to actually exploit the potential monetary and operative cost reduction that is associated to such infrastructures, the amount of resources used by production services must be kept close to the optimum, so the physical resources are exploited as much as possible.

The fast detection of changes in the requirements of the virtual systems deployed on the aforementioned SDN/NFV infrastructures, and the consequent adaptation of allocated resources to the new situations, becomes essential to actually exploit their cost and operative benefits, while also avoiding service unresponsiveness due to underlying resource overloading. It is widely accepted that the size and complexity of systems and services makes it difficult for humans to accomplish such task within their objective time boundaries. Therefore, they must be automated. Luckily, the architecture and underlying platforms supporting the SDN/NFV technologies enable the required automation. In fact, some solutions already exist to perform several batched or scripted tasks without human intervention. However, those solutions still have high dependences on low-level human involvement. This remarks the challenge found in control and management automation, which is continuously revised and enlarged.

ARCA provides as a small step towards the resolution of the aforementioned problem. It advances the State of the Art in automation of resource control and management by providing a supervised but autonomous mechanism that reduces the time required to perform corrective and/or adaptive changes in virtual computer and network systems from hours/minutes to seconds/milliseconds. Moreover, it is able to take advantage of the event notifications provided by external detectors to anticipate the amount of resources that the controlled SDN/NFV system will require in response to such event. We propose to bring such benefit to the reference architecture promoted by ETSI for the management and orchestration of NFV services (see ETSI-NFV-MANO [ETSI-NFV-MANO]) by integrating ARCA as the Virtual Infrastructure Manager (VIM). We showcase this proposal by discussing the evaluation results obtained by ARCA when runnion on a real and physical experimentation infrastructure based on OpenStack [OPENSTACK]. We thus justify the need to adapt the interfaces supported by the NFV-MANO to include real-world event detectors, which are external to the virtualization platform and virtual resources.

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 RFC 2119 [RFC2119].

3. Background

3.1. Virtual Computer and Network Systems

The continuous search for efficiency and cost reduction to get the most optimum exploitation of available resources (e.g. CPU power and electricity) has conducted current physical infrastructures to move towards virtualization infrastructures. Also, this trend enables end systems to be centralized and/or distributed, so that they are deployed to best accomplish customer requirements in terms of resources and qualities.

One of the key functional requirements imposed to computer and network virtualization is a high degree of flexibility and reliability. Both qualities are subject to the underlying technologies but, while the latter has been always enforced to computer and network systems, flexibility is a relatively new requirement, which whould not have been impossed without the backing of virtualization and cloud technologies.

3.2. SDN and NFV

SDN and NFV are conceived to bring high degree of flexibility and conceptual centralization qualities to the network. On the one hand, with SDN, the network can be programmed to implement a dynamic behavior that changes its topology and overall qualities. Moreover, with NFV the functions that are typically provided by physical network equipment are now implemented as virtual appliances that can be deployed and linked together to provide customized network services. SDN and NFV complements to each other to actually implement the network aspect of the aforementioned virtual computer and network systems.

Although centralization can lead us to think on the single-point-of-failure concept, it is not the case for these technologoes. Conceptual centralization highly differs from centralized deployment. It brings all benefits from having a single point of decision but retaining the benefits from distributed systems. For instance, control decisions in SDN can be centralized while the mechanisms that enforce such decisions into the network (SDN controllers) can be implemented as highly distributed systems. The same approach can be applied to NFV. Althoug network functions can be implemented in a central computing facility, they can take advantage of several replication and distribution techniques to achieve the properties of distributed systems. Nevertheless, NFV also allows the deployment of functions on top of distributed systems, so they benefit from both distribution alternatives at the same time.

3.3. Management and Control

The introduction of virtualization into the computer and network system landscape has increased the complexity of both underlying and overlying systems. On the one hand, virtualyzing underlying systems adds extra functions that must be managed propoerly to ensure the correct operation of the whole system, which not just encompasses underlying elements but also the virtual elements running on top of them. Such functions are used to actually host the overlying virtual elements, so there is an indirect management operation that involves virtual systems. Moreover, such complexities are inherited by final systems that get virtualized and deployed on top of those virtualization infrastructures.

In parallel, virtual systems are empowered with additional, and widely exploited, functionality that must be managed correctly. It is the case of the dynamic adaptation of virtual resources to the specific needs of their operation environments, or even the composition of distributed elements across heterogeneous underlying infrastructures, and probably providers.

Taking both complex functions into account, either separately or jointly, makes clear that management requirements have greatly supassed the limits of humans, so automation has become essential to accomplish most common tasks.

3.4. The Autonomic Resource Control Architecture (ARCA)

As deeply discussed in ICIN 2018 [ICIN-2018], ARCA leverages the elastic adaptation of resources assigned to virtual computer and network systems by calculating or estimating their requirements from the analysis of load measurements and the detection of external events. These events can be notified by physical elements (things, sensors) that detect changes on the environment, as well as software elements that analyze digital information, such as connectors to sources or analyzers of Big Data. For instance, ARCA is able to consider the detection of an earthquake or a heavy rainfall to overcome the damages it can make to the controlled system.

The policies that ARCA must enforce will be specified by administrators during the configuration of the control/management engine. Then, ARCA continues running autonomously, with no more human involvement unless some parameter must be changed. ARCA will adopt the required control and management operations to adapt the controlled system to the new situation or requirements. The main goal of ARCA is thus to reduce the time required for resource adaptation from hours/minutes to seconds/milliseconds. With the aforementioned statements, system administrators are able to specify the general operational boundaries in terms of lower and upper system load thresholds, as well as the minimum and maximum amount of resources that can be allocated to the controlled system to overcome any eventual situation, including the natural crossing of such thresholds.

ARCA functional goal is to run autonomously while the performance goal is to keep the resources assigned to the controlled resources as close as possible to the optimum (e.g. 5 % from the optimum) while avoiding service disruption as much as possible, keeping client request discard rate as low as possible (e.g. below 1 %). To achieve both goals, ARCA relies on the Autonomic Computing (AC) paradigm, in the form of interconnected micro-services. Therefore, ARCA includes the four main elements and activities defined by AC, incarnated as:

Collector Is responsible of gathering and formatting the heterogeneous observations that will be used in the control cycle.

Analyzer Correlates the observations to each other in order to find the situation of the controlled system, especially the current load of the resources allocated to the system and the occurrence of an incident that can affect to the normal operation of the system, such as an earthquake that increases the traffic in an emergency-support system, which is the main target scenario studied in this paper.

Decider   Determines the necessary actions to adjust the resources to the load of the controlled system.

Enforcer Requests the underlying and overlying infrastructure, such as OpenStack, to make the necessary changes to reflect the effects of the decided actions into the system.

Being a micro-service architecture means that the different components are executed in parallel. This allows such components to operate in two ways. First, their operation can be dispatched by receiving a message from the previous service or an external service. Second, the services can be self-dispatched, so they can activate some action or send some message without being previously stimulated by any message. The overall control process loops indefinitely and it is closed by checking that the expected effects of an action are actually taking place. The coherence among the distributed services involved in the ARCA control process is ensured by enforcing a common semantic representation and ontology to the messages they exchange.

ARCA semantics are built with the Resource Description Framework (RDF) and the Web Ontology Language (OWL), which are well known and widely used standards for the semantic representation and management of knowledge. They provide the ability to represent new concepts without requiring to change the software, just plugin extensions to the ontology. ARCA stores all its knowledge is stored in the Knowledge Base (KB), which is queried and kept up-to-date by the analyzer and decider micro-services. It is implemented by Apache Jena Fuseki, which is a high-performance RDF data store that supports SPARQL through an HTTP/REST interface. Being de-facto standards, both technologies enable ARCA to be easily integrated to virtualization platforms like OpenStack.

4. External Event Detectors

As mentioned above, current mechanisms used to achieve automated management and control rely only on the continuous monitoring of the resources they control or the underlying infrastructure that host them. However, there are several other sources of information that can be exploited to make the systems more robust and efficient. It is the case of the notifications that can be provided by physical or virtual elements or devices that are watching for specific events, hence called external event detectors.

More specifically, although the notifications provided by these external event detectors are related to successes that occur outside the boundaries of the controlled system, such successes can affect the typical operation of controlled systems. For instance, a heavy rainfall or snowfall can be detected and correlated to a huge increase in the amount of requests experienced by some emergency support service.

5. Anticipating Requirements

One of the main goals of the MANO mechanisms is to ensure the virtual computer and network system they manage meets the requirements established by their owners and administrators. It is currently achieved by observing and analyzing the performance measurements obtained either by directly asking the resources forming the managed system of by asking the controllers of the underlying infrastructure that hosts such resources. Thus, under changing or eventual situations, the managed system must be adapted to cope with the new requirements, incrasing the amount of resources assigned to it, or to make efficient use of available infrastructures, reducing the amount of resources assigned to it.

However, the time required by the infrastructure to make effective the adaptations requested by the MANO mechanisms is longer than the time required by client requests to overload the system and make it discard further client requests. This situation is generally undesired but particularly dangerous for some systems, such as the emergency support system mentioned above. Therefore, in order to avoid the disruption of the service, the change in requirements must be anticipated to ensure that any adaptation has finished as soon as possible, preferably before the target system gets overloaded or underloaded.

Here we propose to integrate ARCA with NFV-MANO to take advantage of the notifications provided by the aforementioned external event detectors, by correlating them to the target amount of resources required by the managed system and enforcing the necessary adaptations beforehand, particularly before the system performance metrics have actually changed.

The following abstract algorithm formalizes the workflow expected to be followed by the different implementations of the operation proposed here.

```
while TRUE do
    event = GetExternalEventInformation()
    if event != NONE then
        anticipated_resource_amount = Anticipator.Get(event)
        if IsPolicyCompliant(anticipated_resource_amount) then
            current_resource_amount = anticipated_resource_amount
            anticipation_time = NOW
        end if
    end if
    anticipated_event = event
    if anticipated_event != NONE and
            (NOW - anticipation_time) > EXPIRATION_TIME then
        current_resource_amount = DEFAULT_RESOURCE_AMOUNT
        anticipated_event = NONE
    end if
    state = GetSystemState()
    if not IsAcceptable(state, current_resource_amount) then
        current_resource_amount = GetResourceAmountForState(state)
        if anticipated_event is not NONE then
            Anticipator.Set
                (anticipated_event, current_resource_amount)
            anticipated_event = NONE
        end if
    end if
end while
```

This algorithm considers both internal and external events to determine the necessary control and management actions to achieve the proper anticipation of resources assigned to the target system. We propose the different implementations to follow the same approach so they can guess what to expect when they interact. For instance, a consumer, such as an Application Service Provider (ASP), can expect some specific behavior of the Virtual Network Operator (VNO) from which it is consuming resources. This helps both the ASP and VNO to properly address resource fluctuations.

6. Information Model

In this section we introduce the basic model needed to support the implementation of the anticipation algorithm. It basically includes the concepts and structures used to describe external events and notify (communicate) them to the interested sink, the network controller/manager, through the control and management plane, depending on the specific instantiation of the system.

6.1. Tree Structure

```
module: ietf-nmrg-nict-resource-anticipation
  +--rw events
     +--rw event-payloads
     +--rw external-events

  notifications:
    +---n event
```

The main models included in the tree structure of the module are the events and notifications. On the one hand, events are structured in payloads and the content of events itself (external-events). On the other hand, there is only one notification, which is the event itself.

6.1.1. event-payloads

```
+--rw event-payloads
   +--rw event-payloads-basic
   +--rw event-payloads-seismometer
   +--rw event-payloads-bigdata
```

The event payloads are, for the time being, composed of three types. First, we have defined the basic payload, which is intended to carry any arbitrary data. Second, we have defined the seismometer payload to carry information about seisms. Third, we have defined the bigdata payload that carries notifications coming from BigData sources.

6.1.1.1. basic

```
+--rw event-payloads-basic* [plid]
   +--rw plid    string
   +--rw data?   union
```

The basic payload is able to hold any data type, so it has a union of several types. It is intended to be used by any source of events that is (still) not covered by other model. In general, any source of telemetry information (e.g. OpenStack controllers) can use this model as such sources can encode on it their information, which typically is very simple and plain. Therefore, the current model is tightly interrelated to a framework to retrieve network telemetry (see [I-D.song-ntf]).

6.1.1.2. seismometer

```
+--rw event-payloads-seismometer* [plid]
   +--rw plid         string
   +--rw location?    string
   +--rw magnitude?   uint8
```

The seismometer model includes the main information related to a seism, such as the location of the incident and its magnitude. Additional fields can be defined in the future by extending this model.

6.1.1.3. bigdata

```
+--rw event-payloads-bigdata* [plid]
   +--rw plid           string
   +--rw description?   string
   +--rw severity?      uint8
```

The bigdata model includes a description of an event (or incident) and its estimated general severity, unrelated to the system. The description is an arbitrary string of characters that would normally carry information that describes the event using some higher level format, such as Turtle or N3 for carrying RDF knowlege items.

6.1.2. external-events

```
+--rw external-events* [id]
   +--rw id           string
   +--rw source?      string
   +--rw context?     string
   +--rw sequence?    int64
   +--rw timestamp?   yang:date-and-time
   +--rw payload?     binary
```

The model defined to encode external events, which encapsulates the payloads introduced above, is completed with an identifier of the message, a string describing the source of the event, a sequence number and a timestamp. Additionaly it includes a string describing the context of the event. It is intended to communicate the required information about the system that detected the event, its location, etc. As the description of the BigData payload, this field can be formated with a high level format, such as RDF.

6.1.3. notifications/event

```
notifications:
  +---n event
     +--ro id?          string
     +--ro source?      string
     +--ro context?     string
     +--ro sequence?    int64
     +--ro timestamp?   yang:date-and-time
     +--ro payload?     binary
```

The event notification inherits all the fields from the model of external events defined above. It is intended to allow software and hardware elements to send, receive, and interpret not just the events that have been detected and notified by, for instance, a sensor, but also the notifications issued by the underlying infrastructure controllers, such as the OpenStack Controller.

6.2. YANG Module

.

```
module ietf-nmrg-nict-resource-anticipation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmrg-nict-resource-anticipation";
  prefix rant;
  import ietf-yang-types { prefix yang; }

  grouping external-event-information {
    leaf id { type string; }
    leaf source { type string; }
    leaf context { type string; }
    leaf sequence { type int64; }
    leaf timestamp { type yang:date-and-time; }
    leaf payload { type binary; }
  }

  grouping event-payload-basic {
    leaf plid { type string; }
    leaf data { type union { type string; type binary; } }
  }

  grouping event-payload-seismometer {
    leaf plid { type string; }
    leaf location { type string; }
    leaf magnitude { type uint8; }
  }

  grouping event-payload-bigdata {
```

```
    leaf plid { type string; }
    leaf description { type string; }
    leaf severity { type uint8; }
  }

  notification event {
    uses external-event-information;
  }

  container events {
    container event-payloads {
      list event-payloads-basic {
        key "plid";
        uses event-payload-basic;
      }
      list event-payloads-seismometer {
        key "plid";
        uses event-payload-seismometer;
      }
      list event-payloads-bigdata {
        key "plid";
        uses event-payload-bigdata;
      }
    }
    list external-events {
      key "id";
      uses external-event-information;
    }
  }

}
```

.

7. ARCA Integration With ETSI-NFV-MANO

In this section we describe how to fit ARCA on a general SDN/NFV underlying infrastructure and introduce a showcase experiment that demonstrates its operation on an OpenStack-based experimentation platform. We first describe the integration of ARCA with the NFV-MANO reference architecture. We contextualize the significance of this integration by describing an emergency support scenario that clearly benefits from it. Then we proceed to detail the elements forming the OpenStack platform and finally we discuss some initial results obtained from them.

7.1. Functional Integration

The most important functional blocks of the NFV reference architecture promoted by ETSI (see ETSI-NFV-MANO [ETSI-NFV-MANO]) are the system support functions for operations and business (OSS/BSS), the element management (EM) and, obviously. the Virtual Network Functions (VNFs). But these functions cannot exist without being instantiated on a specific infrastructure, the NFV infrastructure (NFVI), and all of them must be coordinated, orchestrated, and managed by the general NFV-MANO functions.

Both the NFVI and the NFV-MANO elements are subdivided into several sub-components. The NFVI has the underlying physical computing, storage, and network resources, which are sliced (see[I-D.qiang-coms-netslicing-information-model] and [I-D.geng-coms-architecture]) and virtualized to conform the virtual computing, storage, and network resources that will host the VNFs. In addition, the NFV-MANO is subdivided in the NFV Orchestrator (NFVO), the VNF manager (VNFM) and the Virtual Infrastructure Manager (VIM). As their name indicates, all high-level elements and sub-components have their own and very specific objective in the NFV architecture.

During the design of ARCA we enforced both operational and interfacing aspects to its main objectives. From the operational point of view, ARCA processes observations to manage virtual resources, so it plays the role of the VIM mentioned above. Therefore, ARCA has been designed with appropriate interfaces to fit in the place of the VIM. This way, ARCA provides the NFV reference architecture with the ability to react to external events to adapt virtual computer and network systems, even anticipating such adaptations as performed by ARCA itself. However, some interfaces must be extended to fully enable ARCA to perform its work within the NFV architecture.

Once ARCA is placed in the position of the VIM, it enhances the general NFV architecture with its autonomic management capabilities. In particular, it discharges some responsibilities from the VNFM and NFVO, so they can focus on their own business while the virtual resources are behaving as they expect (and request). Moreover, ARCA improves the scalability and reliability of the managed system in case of disconnection from the orchestration layer due to some failure, network split, etc. It is also achieved by the autonomic capabilities, which, as described above, are guided by the rules and policies specified by the administrators and, here, communicated to ARCA through the NFVO. However, ARCA will not be limited to such operation so, more generally, it will accomplish the requirements established by the Virtual Network Operators (VNOs), which are the owners of the slice of virtual resources that is managed by a particular instance of NFV-MANO, and therefore ARCA.

In addition to the operational functions, ARCA incorporates the necessary mechanisms to engage the interfaces that enable it to interact with other elements of the NFV-MANO reference architecture. More specifically, ARCA is bound to the Or-Vi (see ETSI-NFV-IFA-005 [ETSI-NFV-IFA-005]) and the Nf-Vi (see ETSI-NFV-IFA-004 [ETSI-NFV-IFA-004] and ETSI-NFV-IFA-019 [ETSI-NFV-IFA-019]). The former is the point of attachment between the NFVO and the VIM while the latter is the point of attachment between the NFVI and the VIM. In our current design we decided to avoid the support for the point of attachment between the VNFM and the VIM, called Vi-Vnfm (see ETSI-NFV-IFA-006 [ETSI-NFV-IFA-006]). We leave it for future evolutions of the proposed integration, that will be enabled by a possible solution that provides the functions of the VNFM required by ARCA.

Through the Or-Vi, ARCA receives the instructions it will enforce to the virtual computer and network system it is controlling. As mentioned above, these are specified in the form of rules and policies, which are in turn formatted as several statements and embedded into the Or-Vi messages. In general, these will be high-level objectives, so ARCA will use its reasoning capabilities to translate them into more specific, low-level objectives. For instance, the Or-Vi can specify some high-level statement to avoid CPU overloading and ARCA will use its innate and acquired knowledge to translate it to specific statements that specify which parameters it has to measure (CPU load from assigned servers) and which are their desired boundaries, in the form of high threshold and low threshold. Moreover, the Or-Vi will be used by the NFVO to specify which actions can be used by ARCA to overcome the violation of the mentioned policies.

All information flowing the Or-Vi interface is encoded and formatted by following a simple but highly extensible ontology and exploiting the aforementioned semantic formats. This ensures that the interconnected system is able to evolve, including the replacement of components, updating (addition or removal) the supported concepts to understand new scenarios, and connecting external tools to further enhance the management process. The only requirement to ensure this feature is to ensure that all elements support the mentioned ontology and semantic formats. Although it is not a finished task, the development of semantic technologies allows the easy adaptation and translation of existing information formats, so it is expected that more and more software pieces become easily integrable with the ETSI-NFV-MANO [ETSI-NFV-MANO] architecture.

In contrast to the Or-Vi interface, the Nf-Vi interface exposes more precise and low-level operations. Although this makes it easier to be integrated to ARCA, it also makes it to be tied to specific implementations. In other words, building a proxy that enforces the aforementioned ontology to different interface instances to homogenize them adds undesirable complexity. Therefore, new components have been specifically developed for ARCA to be able to interact with different NFVIs. Nevertheless, this specialization is limited to the collector and enforcer. Moreover, it allows ARCA to have optimized low-level operations, with high improvement of the overall performance. This is the case of the specific implementations of the collector and enforcer used with Mininet and Docker, which are used as underlying infrastructures in previous experiments described in ICIN 2017 [ICIN-2017]. Moreover, as discussed in the following section, this is also the case of the implementations of the collector and enforcer tied to OpenStack telemetry and compute interfaces, respectively. Hence it is important to ensure that telemetry is properly addressed, so we insist in the need to adopt a common framework in such endpoint (see [I-D.song-ntf]).

Although OpenStack still lacks some functionality regarding the construction of specific virtual networks, we use it as the NFVI functional block in the integrated approach. Therefore, OpenStack is the provider of the underlying SDN/NFV infrastructure and we exploited its APIs and SDK to achieve the integration. More specifically, in our showcase we use the APIs provided by Ceilometer, Gnocchi, and Compute services as well as the SDK provided for Python. All of them are gathered within the Nf-Vi interface. Moreover, we have extended the Or-Vi interface to connect external elements, such as the physical or environmental event detectors and Big Data connectors, which is becoming a mandatory requirement of the current virtualization ecosystem and it conforms our main extension to the NFV architecture.

7.2. Target Experiment and Scenario

From the beginning of our work on the design of ARCA we are targeting real-world scenarios, so we get better suited requirements. In particular we work with a scenario that represents an emergency support service that is hosted on a virtual computer and network system, which is in turn hosted on the distributed virtualization infrastructure of a medium-sized organization. The objective is to clearly represent an application that requires high dynamicity and high degree of reliability. The emergency support service accomplishes this by being barely used when there is no incident but also being heavily loaded when there is an incident.

Both the underlying infrastructure and virtual network share the same topology. They have four independent but interconnected network domains that form part of the same administrative domain (organization). The first domain hosts the systems of the headquarters (HQ) of the owner organization, so the VNFs it hosts (servants) implement the emergency support service. We defined them as ‘‘servants’’ because they are Virtual Machine (VM) instances that work together to provide a single service by means of backing the Load Balancer (LB) instances deployed in the separate domains. The amount of resources (servants) assigned to the service will be adjusted by ARCA, attaching or detaching servants to meet the load boundaries specified by administrators.

The other domains represent different buildings of the organization and will host the clients that access to the service when an incident occurs. They also host the necessary LB instances, which are also VNFs that are controlled by ARCA to regulate the access of clients to servants. All domains will have physical detectors to provide external information that can (and will) be correlated to the load of the controlled virtual computer and network system and thus will affect to the amount of servants assigned to it. Although the underlying infrastructure, the servants, and the ARCA instance are the same as those those used in the real world, both clients and detectors will be emulated. Anyway, this does not reduce the transferability of the results obtained from our experiments as it allows to expand the amount of clients beyond the limits of most physical infrastructures.

Each underlying OpenStack domain will be able to host a maximum of 100 clients, as they will be deployed on a low profile virtual machine (flavor in OpenStack). In general, clients will be performing requests at a rate of one request every ten seconds, so there would be a maximum of 30 requests per second. However, under the simulated incident, the clients will raise their load to reach a common maximum of 1200 requests per second. This mimics the shape and size of a real medium-size organization of about 300 users that perform a maximum of four requests per second when they need some support.

The topology of the underlying network is simplified by connecting the four domains to the same, high-performance switch. However, the topology of the virtual network is built by using direct links between the HQ domain and the other three domains. These are complemented by links between domains 2 and 3, and between domains 3 and 4. This way, the three domains have three paths to reach the HQ domain: a direct path with just one hop, and two indirect paths with two and three hops, respectively.

During the execution of the experiment, the detectors notify the incident to the controller as soon as it happens. However, although the clients are stimulated at the same time, there is some delay between the occurrence of the incident and the moment the network service receives the increase in the load. One of the main targets of our experiment is to study such delay and take advantage of it to anticipate the amount of servants required by the system. We discuss it below.

In summary, this scenario highlights the main benefits of ARCA to play the role of VIM and interacting with the underlying OpenStack platform. This means the advancement towards an efficient use of resources and thus reducing the CAPEX of the system. Moreover, as the operation of the system is autonomic, the involvement of human administrators is reduced and, therefore, the OPEX is also reduced.

7.3. OpenStack Platform

The implementation of the scenario described above reflects the requirements of any edge/branch networking infrastructure, which are composed of several distributed micro-data-centers deployed on the wiring centers of the buildings and/or storeys. We chose to use OpenStack to meet such requirements because it is being widely used in production infrastructures and the resulting infrastructure will have the necessary robustness to accomplish our objectives, at the time it reflects the typical underlying platform found in any SDN/NFV environment.

We have deployed four separate network domains, each one with its own OpenStack instantiation. All domains are totally capable of running regular OpenStack workload, i.e. executing VMs and networks, but, as mentioned above, we designate the domain 1 to be the headquarters of the organization. The different underlying networks required by this (quite complex) deployment are provided by several VLANs within a high-end L2 switch. This switch represents the distributed network of the organization. Four separated VLANs are used to isolate the traffic within each domain, by connecting an interface of OpenStack's controller and compute nodes. These VLANs therefore form the distributed data plane. Moreover, other VLAN is used to carry the control plane as well as the management plane, which are used by the NFV-MANO, and thus ARCA. It is instantiated in the physical machine called ARCA Node, to exchange control and management operations in relation to the collector and enforcer defined in ARCA. This VLAN is shared among all OpenStack domains to implement the global control of the virtualization environment pertaining to the organization. Finally, other VLAN is used by the infrastructure to interconnect the data planes of the separated domains and also to allow all elements of the infrastructure to access the Internet to perform software installation and updates.

Installation of OpenStack is provided by the Red Hat OpenStack Platform, which is tightly dependent on the Linux operating system and closely related to the software developed by the OpenStack Open Source project. It provides a comprehensive way to install the whole platform while being easily customized to meet our specific requirements, while it is also backed by operational quality support.

The ARCA node is also based on Linux but, since it is not directly related to the OpenStack deployment, it is not based on the same distribution. It is just configured to be able to access the control and management interfaces offered by OpenStack, and therefore it is connected to the VLAN that hosts the control and management planes. On this node we deploy the NFV-MANO components, including the micro-services that form an ARCA instance.

In summary, we dedicate nine physical computers to the OpenStack deployment, all are Dell PowerEdge R610 with 2 x Xeon 5670 2.96 GHz (6 core / 12 thread) CPU, 48 GiB RAM, 6 x 146 GiB HD at 10 kRPM, and 4 x 1 GE NIC. Moreover, we dedicate an additional computer with the same specification to the ARCA Node. We dedicate a less powerful computer to implement the physical router because it will not be involved in the general execution of OpenStack nor in the specific experiments carried out with it. Finally, as detailed above, we dedicate a high-end physical switch, an HP ProCurve 1810G-24, to build the interconnection networks.

7.4. Initial Results

Using the platform described above we execute an initial but long-lasting experiment based on the target scenario introduced at the beginning of this section. The objective of this experiment is twofold. First, we aim to demonstrate how ARCA behaves in a real environment. Second, we aim to stress the coupling points between ARCA and OpenStack, which will raise the limitations of the existing interfaces.

With such objectives in mind, we define a timeline that will be followed by both clients and external event detectors. It forces the virtualized system to experience different situations, including incidents of many severities. When an incident is found in the timeline, the detectors notify it to the ARCA-based VIM and the clients change their request rates, which will depend on the severity of the incident. This behavior is widely discussed in ICIN 2018 [ICIN-2018], remarking how users behave after occurring a disaster or another similar incident.

The ARCA-based VIM will know the occurrence of the incident from two sources. First, it will receive the notification from the event detectors. Second, it will notice the change of the CPU load of the servants assigned to the target service. In this situation, ARCA has different opportunities to overcome the possible overload (or underload) of the system. We explore the anticipation approach deeply discussed in ICIN 2018 [ICIN-2018]. Its operation is enclosed in the analyzer and decider and it is based on an algorithm that is divided in two sub-algorithms.

The first sub-algorithm reacts to the detection of the incident and ulterior correlation of its severity to the amount of servants required by the system. This sub-algorithm hosts the regression of the learner, which is based on the SVM/SVR technique, and predicts the necessary resources from two features: the severity of the incident and the time elapsed from the moment it happened. The resulting amount of servants is established as the minimum amount that the VIM can use.

The second sub-algorithm is fed with the CPU load measurements of the servants assigned to the service, as reported by the OpenStack platform. With this information it checks whether the system is within the operating parameters established by the NFVO. If not, it adjusts the resources assigned to the system. It also uses the minimum amount established by the other sub-algorithm as the basis for the assignation. After every correction, this algorithm learns the behavior by adding new correlation vectors to the SVM/SVR structure.

When the experiment is running, the collector component of the ARCA-based VIM is attached to the telemetry interface of OpenStack by using the SDK to access the measurement data generated by Ceilometer and stored by Gnocchi. In addition, it is attached to the external event detectors in order to receive their notifications. On the other hand, the enforcer component is attached to the Compute interface of OpenStack by also using its SDK to request the infrastructure to create, destroy, query, or change the status of a VM that hosts a servant of the controlled system. Finally, the enforcer also updates the lists of servers used by the load balancers to distribute the clients among the available resources.

During the execution of the experiment we make the ARCA-based VIM to report the severity of the last incident, if any, the time elapsed since it occurred, the amount of servants assigned to the controlled system, the minimum amount of servants to be assigned, as determined by the anticipation algorithm, and the average load of all servants. In this instance, the severities are spread between 0 (no incident) and 4 (strongest incident), the elapsed times are less than 35 seconds, and the minimum server assignation (MSA) is below 10, although the hard maximum is 15.

With such measurements we illustrate how the learned correlation of the three features (dimensions) mentioned above is achieved. Thus, when there is no incident (severity = 0), the MSA is kept to the minimum. In parallel, regardless of the severity level, the algorithm learned that there is no need to increase the MSA for the first 5 or 10 seconds. This shows the behavior discussed in this paper, that there is a delay between the occurrence of an event and the actual need for updated amount of resources, and it forms one fundamental aspect of our research.

By inspecting the results, we know that there is a burst of client demands that is centered (peak) around 15 seconds after the occurrence of an incident or any other change in the accounted severity. We also know that the burst lasts longer for higher severities, and it fluctuates a bit for the highest severities. Finally, we can also notice that for the majority of severities, the increased MSA is no longer required after 25 seconds from the time the severity change was notified.

All that information becomes part of the knowledge of ARCA and it is stored both by the internal structures of the SVM/SVR and, once represented semantically, in the semantic database that manages the knowledge base of ARCA. Thus, it is used to predict any future behavior. For instance, is an incident of severity 3 has occurred 10 seconds ago, ARCA knows that it will need to set the MSA to 6 servants. In fact, this information has been used during the experiment, so we can also know the accuracy of the algorithm by comparing the anticipated MSA value with the required value (or even the best value). However, the analysis of such information is left for the future.

While preparing and executing the experiment we found several limitation intrinsic to the current OpenStack platform. First, regardless of the CPU and memory resources assigned to the underlying controller nodes, the platform is unable to record and deliver performance measurements at a lower interval than every 10 seconds, so it is currently not suitable for real time operations, which is important for our long-term research objectives. Moreover, we found that the time required by the infrastructure to create a server that hosts a somewhat heavy servant is around 10 seconds, which is too far from our targets. Although these limitations can be improved in the future, they clearly justify that our anticipation approach is essential for the proper working of a virtual system and, thus, the integration of external information becomes mandatory for future system management technologies, especially considering the virtualization environments.

Finally, we found it difficult for the required measurements to be pushed to external components, so we had to poll for them. Otherwise, some component of ARCA must be instantiated along the main OpenStack components and services so it has first-hand and prompt access to such features. This way, ARCA could receive push notifications with the measurements, as it is for the external detectors. This is a key aspect that affects the placement of the NFV-VIM, or some subpart of it, on the general architecture. Therefore, for future iterations of the NFV reference architecture, an integrated view between the VIM and the NFVI could be required to reflect the future reality.

8. Relation to Other IETF/IRTF Initiatives

TBD

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

The major security concerns of the integration of external event detectors and ARCA to manage SDN/NFV systems is that the boundaries of the control and management planes are crossed to introduce information from outside. Such communications must be highly and heavily secured since some malfunction or explicit attacks might compromise the integrity and execution of the controlled system. However, it is up to implementers to deploy the necessary countermeasures to avoid such situations. From the design point of view, since all oprations are performed within the control and/or management planes, the security level of the current solution is inherited and thus determined by the security masures established by the systems conforming such planes.

11. Acknowledgements

TBD

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>.

12.2. Informative References

[ETSI-NFV-IFA-004]
ETSI NFV GS NFV-IFA 004, "Network Functions Virtualisation (NFV); Acceleration Technologies; Management Aspects Specification", 2016.

[ETSI-NFV-IFA-005]
ETSI NFV GS NFV-IFA 005, "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", 2016.

[ETSI-NFV-IFA-006]
ETSI NFV GS NFV-IFA 006, "Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification", 2016.

[ETSI-NFV-IFA-019]
ETSI NFV GS NFV-IFA 019, "Network Functions Virtualisation (NFV); Acceleration Technologies; Management Aspects Specification; Release 3", 2017.

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

[I-D.geng-coms-architecture]
Geng, L., Qiang, L., Lucena, J., Ameigeiras, P., Lopez, D., and L. Contreras, "COMS Architecture", draft-geng-coms-architecture-02 (work in progress), March 2018.

[I-D.qiang-coms-netslicing-information-model]
Qiang, L., Galis, A., Geng, L., kiran.makhijani@huawei.com, k., Martinez-Julia, P., Flinck, H., and X. Foy, "Technology Independent Information Model for Network Slicing", draft-qiang-coms-netslicing-information-model-02 (work in progress), January 2018.

[I-D.song-ntf]
Song, H., Zhou, T., and Z. Li, "Toward a Network Telemetry Framework", draft-song-ntf-01 (work in progress), March 2018.

[ICIN-2017]
P. Martinez-Julia, V. P. Kafle, and H. Harai, "Achieving the autonomic adaptation of resources in virtualized network environments, in Proceedings of the 20th ICIN Conference (Innovations in Clouds, Internet and Networks, ICIN 2017). Washington, DC, USA: IEEE, 2018, pp. 1--8", 2017.

[ICIN-2018]
P. Martinez-Julia, V. P. Kafle, and H. Harai, "Anticipating minimum resources needed to avoid service disruption of emergency support systems, in Proceedings of the 21th ICIN Conference (Innovations in Clouds, Internet and Networks, ICIN 2018). Washington, DC, USA: IEEE, 2018, pp. 1--8", 2018.

[OPENSTACK]
The OpenStack Project, "http://www.openstack.org/", 2018.

Author’s Address

Pedro Martinez-Julia (editor)
NICT
4-2-1, Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Phone: +81 42 327 7293
Email: pedro@nict.go.jp