ETSI GS NFV-INF 001 V1.1.1 (2015-01)

Network Functions Virtualisation (NFV); Infrastructure Overview

Disclaimer

This document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/NFV-INF001

Keywords
architecture, NFV

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

The present document gives an overview to the series of documents covering the NFV Infrastructure.

Infrastructure Architecture Document		Document #
Overview		GS NFV INF 001
Architecture of the Infrastructure Domains	Compute Domain	GS NFV INF 003
	Hypervisor Domain	GS NFV INF 004
	Infrastructure Network Domain	GS NFV INF 005
Architectural Methodology	Interfaces and Abstraction	GS NFV INF 007
Service Quality Metrics		GS NFV INF 010

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**may not**", "**need**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document presents an overview of the architecture of the NFV Infrastructure (NFVI) which supports deployment and execution of Virtualised Network Functions (VNFs).

As well as presenting a general overview description of the NFV Infrastructure, the present document sets the NFV infrastructure and all the documents which describe it in the context of all the documents of the NFV. It also describes how the documents which describe the NFV infrastructure relate to each other.

The present document does not provide any detailed specification but makes reference to specifications developed by other bodies and to potential specifications, which, in the opinion of the NFV ISG could be usefully developed by an appropriate Standards Developing Organisation (SDO).

The overall objectives of the ISG NFV were set out in the white paper [i.1] that led to the founding of the ISG and updated in the white paper update [i.2].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI ETSI GS NFV 001 (V1.1.1) (10-2013): "Network Functions Virtualisation (NFV); Use Cases".

[2] ETSI ETSI GS NFV 002 (V1.1.1) (10-2013): "Network Functions Virtualisation (NFV); Architectural Framework".

[3] ETSI ETSI GS NFV 003 (V1.1.1) (10-2013): "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[4] ETSI ETSI GS NFV 004 (V1.1.1) (10-2013): "Network Functions Virtualisation (NFV); Virtualisation Requirements".

[5] ETSI GS NFV-PER 002 (V1.1.1) (10-2013): "Network Functions Virtualisation (NFV); Proofs of Concepts; Framework".

[6] ETSI GS NFV-SEC 001 (V1.1.1) (10-2014): "Network Functions Virtualisation (NFV); NFV Security; Problem Statement".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] NFV Whitepaper: "Network Function Virtualization", issue 1, (2012).

NOTE: Available at http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[i.2] NFV Whitepaper: "Network Function Virtualization - Update White Paper", issue 2, (2013).

NOTE: Available at http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

[i.3] IEEE Cloud Computing 2009: "The Method and Tool of Cost Analysis for Cloud Computing", Ying Li, Tiancheng Liu, Jie Qiu, Fengchun Wang.

[i.4] IEEE System Science (HICSS) (2012): "Costing of Cloud Computing Services: A Total Cost of Ownership Approach", B. Martens, M. Walterbusch, F. Teuteberg.

[i.5] TR174 Enterprise-Grade IaaS Requirements Rev1.3.

NOTE: Available at http://www.tmforum.org/TechnicalReports/EnterpriseGradeExternal/50445/article.html.

[i.6] The Open Virtualization Format (OVF) Specification, Version 2.0, 2012, Distributed Management Task Force.

NOTE: Available at http://dmtf.org/sites/default/files/standards/documents/DSP0243_2.0.0.pdf.

[i.7] Master Usage Model: Compute Infrastructure as a Service, Rev 1, (2012), Open Data Center Alliance.

NOTE: Available at http://www.opendatacenteralliance.org/docs/ODCA_Compute_IaaS_MasterUM_v1.0_Nov2012.pdf.

[i.8] Usage Model: Guide to Interoperability Across Clouds, 2012, Open Data Center Alliance.

NOTE: Available at http://www.opendatacenteralliance.org/docs/ODCA_Interop_Across_Clouds_Guide_Rev1.0.pdf.

[i.9] USAGE: Input/Output (IO) Controls , Rev 1.1., 2012, Open Data Center Alliance.

NOTE: Available at http://www.opendatacenteralliance.org/docs/IO_Controls_Rev_1.1_b.pdf.

[i.10] NIST SP-800-145 (September 2011): "The NIST Definition of Cloud Computing," Peter Mell and Timothy Grance, US National Institute of Standards and Technology.

NOTE: Available at http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[i.11] Recommendation ITU-T Q.1741: "GSM evolved UMTS core network".

[i.12] ETSI GS NFV-INF 003: "Network Functions Virtualisation (NFV); Infrastructure; Compute Domain".

[i.13] ETSI GS NFV-INF 004: "Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain".

[i.14] ETSI GS NFV-INF 005: "Network Functions Virtualisation (NFV); Infrastructure; Infrastructure Network Domain".

[i.15] ETSI GS NFV-INF 007: "Network Functions Virtualisation (NFV); Infrastructure; Methodology to describe Interfaces and Abstractions".

[i.16] A. Capiluppi, K-J.Stol, C. Boldyreff, "Software reuse in Open Source: A Case Study", Int"l J. of Open Source Software and Processes, Vol 3. Iss. 3, (2011).

[i.17] NIST SP-800-146: "Cloud Computing Synopsis and Recommendations".

[i.18] IEEE 802.1Q™: "Virtual LANs".

[i.19] IEEE 802.1ad™: "Support on Provider Bridges".

[i.20] ISO/IEC JTC1 SC 38: "Distributed application platforms and services (DAPS)".

NOTE: Available at http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/jtc1_home/jtc1_sc38_home.htm.

[i.21] Recommendation ITU-T SG13: "Future networks including cloud computing, mobile and next-generation networks".

NOTE: Available at http://www.itu.int/en/ITU-T/studygroups/2013-2016/13/Pages/default.aspx.

[i.22] Recommendation ITU-T SG15: "Networks, Technologies and Infrastructures for Transport, Access and Home".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

container interface: environment within a HFB which is configured in or to realize a VFB

NOTE 1: This includes the configurability and/or programming language of the environment. The container interface is ***not*** an interface between functional blocks.

NOTE 2: Container interface should not be confused with 'containers' as used in the context of Unix type operating systems as an alternative to full virtual machines.

NOTE 3: The relation between a container interface as defined in the present document and a virtualization container as defined in the ETSI GS NFV 003 [3] is for further study.

domain: specific part of a larger entity which is useful to separate out based on given criteria

NOTE: Domains can be defined for many different purposes and the features which distinguish domains may differ in different contexts.

EXAMPLE: The compute domain, hypervisor domain, and infrastructure network domain may not be administrative domains.

functional block: basis element of a system

NOTE: A Functional Block has interfaces (both input interfaces, output interfaces), can hold state, and evolves its state and output parameters according to a unchanging transfer function.

Host Functional Block (HFB): functional block which can be configured and/or programmed

NOTE: When suitable configured and/or programmed, a Host Function Block behaves as if it were one or more functional blocks with a more specific definition. A Host Functional Block is said to host one or more Virtual Functional Blocks.

hypervisor: piece of software which partitions the underlying physical resources and creates Virtual Machines, and isolates the VMs from each other

NOTE: The Hypervisor is a piece of software running either directly on top of the hardware (bare metal hypervisor) or running on top of a hosting operating system (hosted hypervisor). The abstraction of resources comprises all those entities inside a computer/server which are accessible, like processor, memory/storage, NICs. The hypervisor enables the portability of VMs to different Hardware.

infrastructure interface: interface between two HFBs

NOTE: An Infrastructure Interface can transport a virtualised interface without placing any dependency on the particular type of virtualised interface.

Network Element (NE): discrete telecommunications entity, which can be managed over a specific interface, e.g. the RNC (from Recommendation ITU-T Q.1741 [i.11])

Network Function (NF): Functional Block (FB) within a network infrastructure which has well-defined external interfaces and well-defined functional behaviour

NOTE: In practical terms, a Network Function is today often a network node or physical appliance.

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

NOTE: The NFV-Infrastructure can span across several locations, e.g. places where data centres are operated. The network providing connectivity between these locations is regarded to be part of the NFV-Infrastructure. NFV-Infrastructure and VNF are the top-level conceptual entities in the scope of Network Function Virtualization. All other components are sub-entities of these 2 main entities.

NFVI-Node: physical device deployed and managed as a single entity providing the NFVI functions required to support the execution environment for VNFs

NFVI-PoP: single geographic location where a number of NFVI-Nodes are sited

portability: ability to transfer data from one system to another without being required to recreate or re-enter data descriptions or to modify significantly the application being transported

Virtual Functional Block (VFB): functional block, defined in a logical, implementation independent way, which is implemented by configuring a host functional block

NOTE: Programming is a form of configuration.

Virtual Machine (VM): virtualized computation environment which behaves very much like a physical computer/server

NOTE: A VM has all its ingredients (processor, memory/storage, interfaces/ports) of a physical computer/server and is generated by a Hypervisor, which partitions the underlying physical resources and allocates them to VMs. Virtual Machines are capable of hosting a VNF Component (VNFC).

virtualized interface: interface, defined in a logical and abstract way, between two VFBs

virtual network: topological component used to affect forwarding of specific characteristic information

NOTE 1: The virtual network is bounded by its set of permissible network interfaces.

NOTE 2: In the NFVI architecture, a virtual network forwards information among the network interfaces of VM instances and physical network interfaces, providing the necessary connectivity and ensures secure isolation of traffic from different virtual networks.

Virtualised Network Function (VNF): implementation of an NF that can be deployed on a Network Functions Virtualisation Infrastructure (NFVI)

NOTE: A VNF is a VFB which provides exactly the same functional behaviour and interfaces as the equivalent Network Function.

Virtualised Network Function Component (VNFC): internal component of a VNF providing a VNF Provider-defined sub-set of that VNF's functionality, with the main characteristic that a single instance of this component maps 1:1 against a single VM Container Interface

NOTE: A VNFC which has been instantiated and deployed in a VM is called a VNFC Instance. A VNFC which is part of the resource pool is called a VNFC Resource, and a reserved VNFC is called a Reserved VNFC Resource. A more general VNF may be a functional composition of a number of VNFCs.

The following definitions relate to the specific domains of the NFVI, the compute domain, the hypervisor domain, and the infrastructure network domain. Further definitions relating to each domain are contained in each respective domain architecture document.

accelerator: co-processor or other specialized hardware entity deployed to offload processing, or otherwise improve performance of software running on a main processor

Central Processing Unit (CPU): device in the compute node which provides the primary container interface

NOTE: The CPU instruction set is the primary runtime and execution language of the compute node. A programme of CPU instructions loaded into memory and executing is the primary way by which a compute node acts as a HFB and hosts VFBs. A specific VFB is defined by the specific programme for that VFB running on the specific CPU.

compute domain: general area for focus which includes servers and storage

NOTE: The compute domain has its own architecture documentation within the Infrastructure architecture.

compute Nnde: single identifiable, addressable, and manageable element within an NFVI-Node that provides computing resource using compute, storage, and networking functions

NOTE: A Compute Node is normally programmable and can run a hypervisor which supports VM instances. Stand-alone acceleration devices are also compute nodes.

execution cycle: step in the evolution of state within a compute node

NOTE: Strictly, this can be defined abstractly, in practical terms, this will relate directly to a CPU clock cycle.

gateway node: single identifiable, addressable, and manageable element within an NFVI-Node that implements gateway functions

hypervisor domain: general area for focus which includes hypervisors

infrastructure connectivity service: connectivity service provided by the infrastructure network domain

NOTE: The Infrastructure Connectivity Services abstract the details of topology, switching equipment, and protocol/encapsulations of the infrastructure network domain. In practice relevant examples of Infrastructure Connectivity Service as likely to include E-Line and E-LAN services as defined by Metro-Ethernet Forum (MEF).

infrastructure network domain: general area for focus which includes all networking which interconnects compute/storage infrastructure and pre-exists the realisation of VNFs

Network Interface Controller (NIC): device in a compute node which provides a physical interface with the infrastructure network

network node: single identifiable, addressable, and manageable element within an NFVI-Node that provides networking (switching/routing) resource using compute, storage, and network forwarding functions

NOTE: This is a node in the NFV Infrastructure network and if the context is not clear should be called an Infrastructure Network Node.

offload: delegating processing (e.g. classification, forwarding, load balancing, cryptography, transcoding) to a different processor or other specialized hardware entity

state: set of all parameters held within a functional block

NOTE: For a compute node, in practice, this is the memory (volatile and non-volatile).

storage: non-volatile storage with in the compute domain

NOTE: In practice, this is likely to be implemented as spinning disks or as solid-state disks.

storage node: single identifiable, addressable, and manageable element within an NFVI-Node that provides storage resource using compute, storage, and networking functions

vCPU: virtualised CPU created for a VM by a hypervisor

NOTE: In practice, a vCPU may be a time sharing of a real CPU and/or in the case of multi-core CPUs, it may be an allocation of one or more cores to a VM. It is also possible that the hypervisor may emulate a CPU instruction set such that the vCPU instruction set is different to the native CPU instruction set (emulation will significantly impact performance).

vNIC: virtualised NIC created for a VM by a hypervisor

vStorage: virtualised non-volatile storage allocated to a VM

vSwitch: Ethernet switch implemented in the hypervisor domain which interconnects vNICs of VMs with each other and with the NIC of the compute node

NOTE: The vSwitch may be combined with the hypervisor as a single software package or provided as a standalone piece of software running on top of or aside the hypervisor.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP	3rd Generation Partnership Project
ACL	Access Control List
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
ATIS	Alliance for Telecommunications Industry Solutions
BIOS	Basic Input Output System
BSS	Business Support System
CDN	Content Distribution Network
COTS	Commercial Off The Self
CPU	Central Processor Unit
CSA	[TBC by EG SEC]
CSCF	Call Session Control Function
DMA	Direct Memory Access
DMTF	Distributed Management Task Force
DNS	Domain Name System
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DRAM	Dynamic Random Access Memory
E-LAN	Ethernet Local Area Network
E-Line	Ethernet Line
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FB	Functional Block
FIFO	First In First Out
GW	Gateway
HFB	Host Functional Block
HSS	Home Subscriber Server
HW	Hardware
IaaS	Infrastructure as a Service
I-CSCF	Interrogating Call Session Control Function
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
INCOSE	International Council on Systems Engineering
ISG	Industry Specification Group

ISO	International Standard Organisation
IT	Information Technology
ITU-T	International Telecommunications Union - Telecommunications
KLR	Kernel Level Rootkits
KQI	Key Quality Indicators
KVM	Kernel-Based Virtual Machine
LAN	Local Area Network
MAC	Media Access Control
MANO	Management and Orchestration
MEF	Metro Ethernet Forum
MME	Mobility Management Entity
NAPI	Network Application Programming Interface
NE	Network Element
NF	Network Function
NFV	Network Functions Virtualisation
NFVI	Network Functions Virtualisation Infrastructure
NFVI-Node	Network Functions Virtualisation Infrastructure Node
NFVI-PoP	Network Functions Virtualisation Infrastructure Point of Presence
NFVO	Network Functions Virtualisation Orchestration
NIC	Network Interface Card
NIST	National Institute for Standards and Technology
NSI	North American Association of State and Provincial Lotteries Standards Initiative
NVF	Network Functions Virtualisation
OAM	Operations, Administration, and Maintenance
OASIS	Organization for the Advancement of Structured Information Standards
OCP	Open Computing Project
ODCA	Open Data Centre Alliance
OMG	Object Management Group
ONF	Open Networking Foundation
OS	Operating System
OSS	Operational Support System
PaaS	Platform as a Service
PCEF	Policy and Charging Enforcement Function
PCIe	Peripheral Component Interconnect - Express
PCRF	Policy and Charging Rule Function
P-CSCF	Proxy Call Session Control Function
P-GW	Proxy Gateway
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
SaaS	Software as a Service
SAN	Storage Area Network
S-CSCF	Serving Call Session Control Function
SDN	Software-Defined Networking
SDN-C	Software Defined Networking - Controller
SDO	Standards Developing Organisation
S-GW	Serving Gateway
SLA	Service Level Agreement
SNIA	Storage Networking Industry Association
SR-IOV	Single Root Input Output Virtualisation
SSD	Solid State Disk
SSH	Secure Shell
SSO	Standards Setting Organisation
SW	Software
SWA	SoftWare Architecture
TCP	Transport Control Protocol
TLS	Transport Layer Security
TMF	TM Forum
UML	Unified Modelling Language
vCPU	Virtual Central Processor Unit
VFB	Virtual Functional Block
VFND	Virtualised Network Function Descriptor
VIM	Virtual Infrastructure Manager

VLAN	Virtual Local Area Network
VM	Virtual Machine
VN	Virtual Network
VNF	Virtualised Network Function
VNFC	Virtualised Network Function Component
VNFD	Virtualised Network Function Descriptor
vNIC	Virtual Network Interface Card
XML	eXtensible Markup Language

4 Objectives of the NFV Infrastructure

The objectives of the Network Function Virtualisation Infrastructure emerge from consideration of the overall objectives for NFV and the role that the NFVI has in supporting the NFV ecosystem. The NFV Use Cases document [1] identifies 9 fields of application or use cases for NFV. The NFVI is the totality of the hardware and software components which build up the environment in which VNFs are deployed [3]. The NFVI is deployed as a distributed set of NFVI-nodes in various locations to support the locality and latency requirements of the different use cases and the NFVI provide the physical platform on which the diverse set of VNFs are executed; enabling the flexible deployment of network functions envisaged by the NFV Architectural Framework [2].

From a functional perspective, the NFVI provide the technology platform with a common execution environment for the NFV use cases as illustrated in figure 1. The NFVI provide the infrastructure that support one or more of these use cases simultaneously and is dynamically reconfigurable between these use cases through the installation of different VNFs.

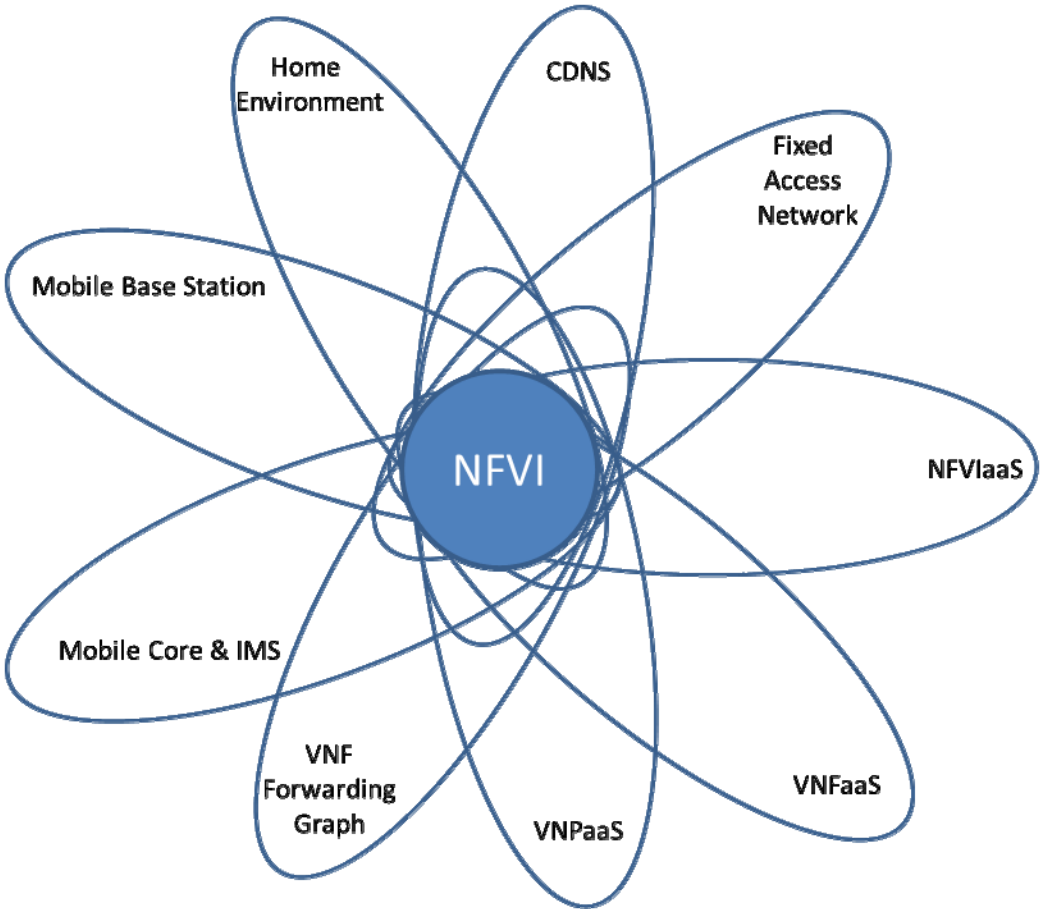

Figure 1: NFVI as the execution environment for NFV Use cases

The introductory NFV Whitepaper [i.1] envisaged an NFV ecosystem offering integration services as well as maintenance and third party support. The 2013 White paper update [i.2] reiterated the need for encouraging the development of an open NFV ecosystem with industry capabilities for the integration of solutions from different technology providers; where this open and innovative NFV ecosystem supported the porting of VNFs between nodes of the NFVI. The PoC Framework [5] was developed to further encourage this open global NFV ecosystem through the integration of components from different technology providers. The NFVI is the technology platform enabling the open innovation expected in the NFV ecosystem.

The NFV Architectural Framework [2] provides some initial description of the NFVI in clause 7.2.4 of that specification. The NFV Requirements Specification [4] identifies a number of requirements for an NFVI to fulfil the role of supporting the execution environment for VNFs. These requirements span areas such as security, Operations and Maintenance and Service Models.

4.1 Standardizing Organisations Impacting the NFVI

The ETSI NFV ISG is not intended to be a Standards Development Organization (SDO) (see [i.1], page 13); rather the documentation produced by the ISG, including this specification, should reference the work of existing SDOs and identify gaps where additional standardization efforts by those SDOs may be required to further enable the NFV ecosystem.

There are many standards organisations which already have or are progressing standards which are relevant to the NFVI and these are referenced at the end of each clause of the present document. At the general level, NIST is responsible for developing terminology for cloud computing applications and infrastructure. Cloud computing is an enabling technology for network function virtualization. The relationship between cloud computing and NFV is discussed further in clause 6.2.

The 2013 Operator whitepaper update [i.2] also identifies the relevance of open source approaches as complementary to formal standardization efforts in the development of reference implementations as a substrate for open innovation in the NFV ecosystem. Open source developer communities and user communities can be helpful in the further development of the NFV ecosystem, (e.g. open source communities can provide a mechanism for increased software reuse [i.16]; and software reuse may increase the efficiency in the NFV ecosystem, as well as enabling more rapid development and deployment of new services which would support the increased service velocity objective of NFV. Open Source Communities do exist in support of projects relevant to the NFVI (e.g. the XEN and KVM hypervisors). The NFVI may include open source software components, but the role and collaboration mechanisms of open source communities in the NFV ecosystem are beyond the scope of the present document.

5 Structure of NFV Infrastructure Architecture Documentation

The overall architectural framework for NFV is set out in the "Architectural Framework" document [2]. Figure 4 of that document provides the NFV reference architectural framework and shows for the NFV Infrastructure (NFVI) interactions with other parts of the NFV architecture, notably the Virtual Network Function (VNF) architecture (defined by the SWA Working Group), and the Management and Orchestration architecture (defined by the MANO Working Group) and well as with existing networks and systems.

NOTE 1: The Architectural Framework [2] is one of four documents covering the overall scope of ISG NFV. The others documents cover Use Cases [1], Terminology [3], and Requirements [4].

The architecture of the NFV Infrastructure (NFVI) is further separated into three domains, namely the compute domain (including storage), the hypervisor domain, and the infrastructure network domain. These domains are largely distinct both at a functional and practical level (see note). The general positioning of these domains within the NFV reference architectural framework is shown in figure 2.

NOTE 2: Examples of overlap functionality are a virtual Ethernet switch (vSwitch) and an embedded Ethernet switch in a NIC. These are discussed in more detail in clause 6.

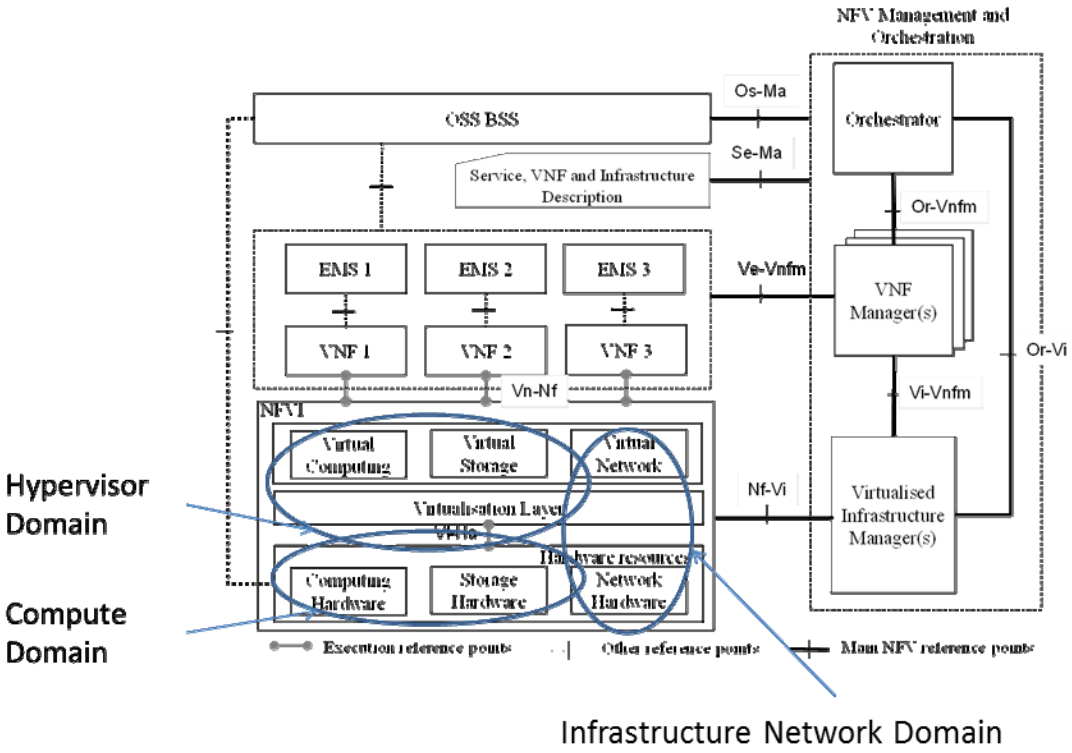

Figure 2: NFV reference architectural framework and identification of NFVI domains

The documentation of the NFVI Architecture is structured as follows:

- Overview Documents. These documents give the overview and set out the feature for the Network Functions Virtualisation Infrastructure (NFVI):
  - Overview of the Architecture of the NFVI (the present document).
  - Use Cases of the NFVI.
- Domain Documents. These documents define the architecture of the three domains of the NFVI:
  - Compute Domain Architecture.
  - Hypervisor Domain Architecture.
  - Infrastructure Network Architecture.
- Methodology Documents. These documents set out principles for the specification, construction, and exploitation of the NFVI:
  - Interfaces and Abstraction. The present document set out some of the foundations of virtualisation. In particular, it describes the way interfaces are specified, the way practical interfaces use an abstract view of a function, and the way in which a VNF is always a abstract view of a network function.
  - Portability and Replicability. The present document set out the way in which the portability of VNFs between different implementation of NFVI may be assured. In addition, the document set out more generally the way one implementation of a component either of a VNF or of the NFVI may be replicated and replaced with another implementation. The present document provides the framework by which economic analysis including business case analysis can be carried out.

The primary specification of the NFVI is therefore contained in the three domain architecture documents. The methodology documents contain important information about how the NFVI can be used in order to achieve the benefits set out in the white paper.

This NFVI Architecture Overview is structured as follows:

- Clause 6 sets out a number of important architectural principles on which the NFVI architecture is based. It covers a number of underlying foundations which are relied on in developing the more specific architecture of the NFVI including:
  - virtualisation and functional block methodology (clause 6.1);
  - how NFV builds on Cloud Computing (clause 6.2);
  - the principle of domains and the identification of the domains of the NFVI (clause 6.3);
  - the principles and the multiplicities in mapping VNFs to host functions in the NFVI (clause 6.4);
  - economic and practical issues in achieving interoperability (clause 6.5);
  - key quality indicators (KQIs) for the NFVI (clause 6.6);
  - overview of security aspects affecting the NFVI (clause 6.7).
- Clause 7 sets an overview of the architecture of each of the NFVI domains:
  - compute domain (clause 7.1);
  - hypervisor domain (clause 7.2);
  - infrastructure network domain (clause 7.3).
- Clause 8 sets out an overview of the performance challenges for the NFVI.

Some of the principles set out in clause 6 are defined in more detail in the NFVI documents "Methodology to describe: Interfaces and Abstractions" (ETSI GS NFV-INF 007 [i.15]).

Clause 7 gives an overview of the architecture of each NFVI domain. The detailed architecture of each domain are given in "NFV Infrastructure Architecture: Compute Domain" (ETSI GS NFV-INF 003 [i.12]), "NFV Infrastructure Architecture: Hypervisor Domain" (ETSI GS NFV-INF 004 [i.13]), and "NFV Infrastructure Architecture: Infrastructure Network Domain" (ETSI GS NFV-INF 005 [i.14]) respectively.

6 Architectural Principles of the Network Functions Virtualisation Infrastructure (NFVI)

This clause sets out an overview of a number of central principles on which the NFVI architecture is based.

6.1 Virtualisation and Associated Interfaces

This clause provides an overview of the way functional block methodology is developed to describe, define, and specify virtualisation of functions. Details of the way functional block methodology is used to describe, define, and specify virtual functions such as VNFs are set out in "Methodology to describe: Interfaces and Abstractions" (ETSI GS NFV-INF 007 [i.15]).

6.1.1 Describing and Specifying Network Functions When Virtualised

Many network systems, including those specified by 3GPP, IETF, and ITU-T, are specified using the principles of systems engineering. Each component of the overall system is specified as a functional block and the interactions between the functional blocks are specified as interfaces. A significant, fundament principle of the systems engineering approach to specification is that the complete system is specified by the specification of the component functional blocks and their interconnection, and all components are always functional blocks.

A traditional functional block diagram is illustrated in figure 3.

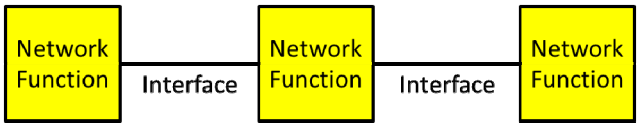

Figure 3: Functional Block Architecture

An inherent property of a functional block is that its operation is autonomous. The behaviour of the function block is determined by:

- The static transfer function of the functional block.
- The dynamic state of the functional block.
- The inputs it received in its interfaces.

If a functional block becomes disconnected from a functional block to which it should be connected, it will continue executing and setting outputs, however, the execution will follow according to the null input.

As stated above, the objective of NFV is to separate software that defines the network function (the VNF) from hardware and generic software that create a generic hosting network functions virtualisation infrastructure (NFVI) which executes the VNF. It is therefore a requirement that the VNFs and the NFVI be separately specified. However, this is a requirement that is not immediately satisfied by the method of functional blocks and associated interfaces a generalization of the technique.

Figure 4 shows the situation when two of the three functional blocks of figure 3 have been virtualised. In each case, the functional block is now implemented as a virtualised network function executing on a host function in the NFVI.

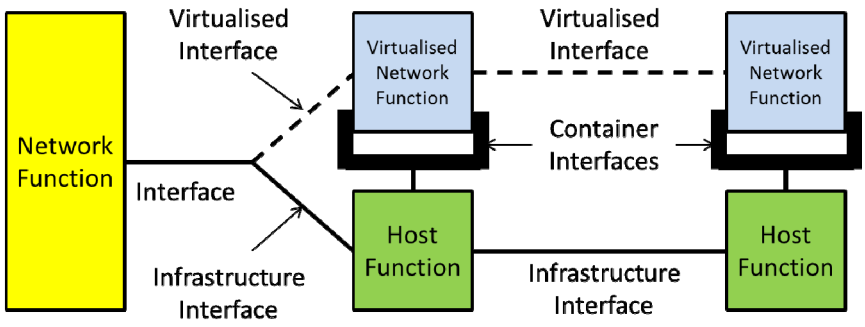

Figure 4: Virtualisation of network function

This process has resulted in the following:

- The division of a functional block between a host function and a virtualised network function.
- The creation of a new container interface between the host function and the VNF it is hosting.
- The division of the interface between the two network functions which are now virtualised between an infrastructure interface and a virtualised interface.
- The interface to the non-virtualised network function appears to be a homogeneous interface at its end with the non-virtualised function. However, at its virtualised end, it appears to be divided between an infrastructure interface and a virtualised interface.

However, in carrying out this process, there are two very important distinctions from the standard functional block description:

- The virtualised network function (VNF) is ***not*** a functional block independent of it host function.
- The container interface is ***not*** an interface between functional blocks equivalent to other interfaces.

This arises as the VNF cannot exist autonomously in the way a functional block can. The VNF depends on the host function for its existence, and if the host function were to be interrupted, or even disappear, then the VNF will also be interrupted or disappear. Likewise, the container interface reflects this existence dependency between a VNF and its host function.

The relationship between the VNF and its host function is:

- the VNF is a configuration of the host function.
- the VNF is an abstract view of the host function when the host function is configured by the VNF.

Therefore a host function, when configured with a VNF, has the ***external appearance*** as being a functional block implementing the VNF specification. It is the host function that is the functional block but it appears from the outside to be the VNF. Equivalently, the VNF is an abstract view of the host function.

A more formal description of this relationship is set out in clause 6.1.3.

The NFV architecture is therefore defined using not just functional blocks and their associated interfaces, but is defined using the following entities:

- Host functions with their associated offered container interfaces and associated infrastructure interfaces.
- Virtualised Network Functions (VNFs) with their associated used container interfaces and virtualised interfaces.

The NFV documentation uses these constructs. The NFV documentation also noted that the container relationship between host function and virtualised function can be recursive layered. Host functions with more and more specific capability can be built up by the successive configuration with layers of virtual functions.

A specific example which is especially pertinent to NFV is as follows:

- A base host function which is a server blade (a specific implementation of a compute node, see below). This provides a container interface of extreme generality which is the 'bare metal' machine interface and the associated BIOS.
- This base host function is then configured with a virtual function (a programme is loaded and executed) which is a hypervisor. The server blade now configured with the hypervisor can provide a number of virtual machine container interfaces.
- A virtual machine can be configured with a virtual function (a programme which is loaded and executed) which is an operating system. The server blade now configured with the hypervisor and guest operating can provide a number of application container interfaces.
- A specific application can be configured with a virtual function (a programme which is loaded and executed) which is the application. The application itself may be capable of many different functional operations depending on its configuration.
- A specific functional operation can be configured with a virtual function which a specific configuration of the application.

6.1.2 NFV Interoperability Challenges

Interoperability challenges typically arise where interfaces cross administrative boundaries. The use of NFV within an administrative region does not present any new interoperability challenges between regions when the regions are interconnected by existing network function interfaces.

Consider the types of interfaces identified in figure 4: Existing Network Function Interfaces, Infrastructure Interfaces, Container Interfaces, and Virtualized Interfaces.

Existing Network Function Interfaces are already specified by other Standards Setting Organizations (SSOs). Those SSOs are already engaged in the process of identifying and resolving interoperability issues.

Infrastructure Interfaces are largely specified by the IT industry for cloud infrastructures. The IT industry is already engaged with resolving any Interoperability concerns with those interfaces.

Container interfaces have been specified by the IT industry for a number of applications, typically web service based. Extensions of these interfaces to support virtualized network functions may present additional interoperability challenges. Interoperability challenges exist here because the NFV concept assumes a multivendor environment supplying VNF and infrastructure components.

Any interoperability challenges with virtualized interfaces that are the functional equivalents of existing network function interfaces would likely be resolved by the SSOs that developed the original network function interfaces as they have the necessary deep functional expertise. Novel virtualization interfaces that do not correspond to existing functional interfaces (e.g. due to some form of disaggregated virtualization) may present novel interoperability challenges, perhaps motivating the participants involved to prefer more standard solutions.

6.1.3 Management and Orchestration When Network Functions are Virtualised

In a telecommunications operator's environment, network functions are capable of remote configuration and management. In order to achieve this, the network functions have an interface (often referred to as a 'north-bound' interface) to a management and orchestration function. This management and orchestration function is often highly complex and composed of a great many distributed component parts. However, it is still defined using the same systems engineering techniques of functional blocks and interfaces. This is illustrated in figure 5.

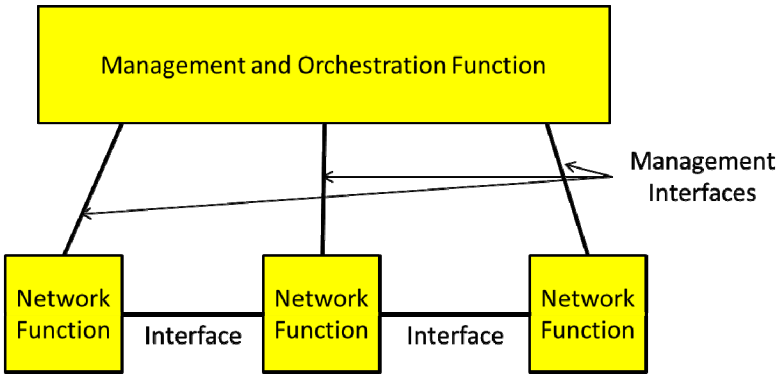

Figure 5: Management and orchestration of network functions

The objective of NFV is to separate the virtualised network functions from the infrastructure, and this includes their management. As shown in figure 6, the management and orchestration functions are divided between the management and orchestration of the network functions virtualisation infrastructure (NFVI) and the management and orchestration of the VNFs.

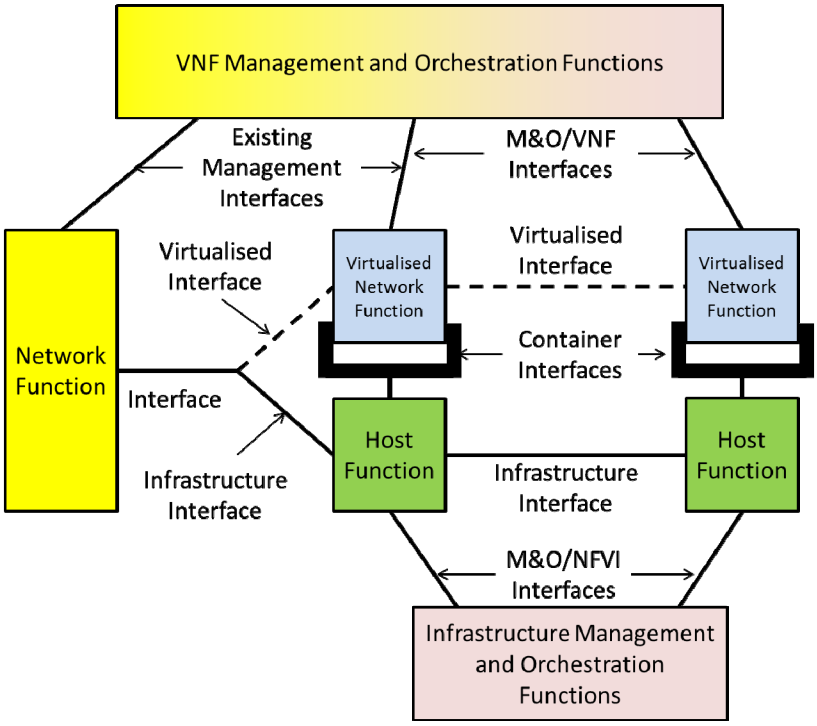

Figure 6: Management and orchestration of virtualised network functions

The management and orchestration of the NFVI is an integral and essential part of the NFV framework and is specified within the GS NFV MANO documentation.

It is likely that there is existing management of the network functions that are now virtualised by a NFV deployment. Managing the VNFs using the existing systems can be used for the deployment of NFV. This is represented by the yellow in the VNF Management and Orchestration Functions block in figure 6.

On the other hand, removal of the hardware from the VNFs removes the requirement to manage hardware aspects. In addition, the great flexibility offered by NFV can only be fully exploited if the management and orchestration implements efficient VNF lifecycle management process adapted to new business process requirements: fast order delivery, fast recovery, auto-scaling, etc. Therefore, the target objective is that the management and orchestration of VNFs is that they can be managed and orchestrated using high levels of common management and orchestration functionality with a largely common management and orchestration interface specification. This is represented by the pink in figure 6.

6.1.4 Brief Formal Description and Definition of Virtualisation

NFV brings together a number of different industry sectors including network design and operations, network equipment, compute equipment, hypervisors and cloud technologies, as well as management, orchestration, and other operational and business support systems. Each of these has developed a set of descriptive techniques and languages that are used in specification and a generally understood within the industry without further explanation. However, in the NFV two problems arise:

- practitioners from one industry may not be familiar with the techniques and languages from other industries;
- the techniques and languages from any one industry may not be sufficient for the needs of the other industries.

The formal specification methodology used in NFV is that of systems engineering using functional blocks.

However, while there is considerable science and precision to the methodology of functional blocks, there is a major ambiguity with the representation of virtualisation using functional blocks. The ambiguity is this:

- the methodology represents everything as functional blocks interconnected by interfaces;
- the relationship between a hosting functional block and a hosted virtual functional block hosted is not that of an interface.

In order to set out the technical mechanism of virtualisation in the methodology of functional blocks, it is necessary to start with the essential characteristics of a functional block. A functional block comprises:

- inputs interfaces through which interconnected functional blocks select an input value form a predefined range of possible input values;
- internal state which has a value from a predefined range of values and which is set by the internal operation of the functional block (state holds the history of inputs as this affects further evolution of the functional block);
- output interfaces on which the functional block selects a value from a predefined range of values which then selects the input values on interconnected functional blocks;
- a transfer function which is a predefined transition matrix which mapped a specific tuple of input and current state to a specific value of next state and output.

These characteristics are all necessary for the complete specification of a functional block, and when taken together, are also sufficient for complete specification.

The interesting property from the perspective of virtualisation is the things that are predefined. The value ranges of the inputs, outputs, and state are all predefined, as is the transfer function. These are static to the functional block and defining of the functional block. Defining these static parameters fully specifies the functional block. The specific values of the inputs, outputs, and state are dynamic to the functional block and are not part of its specification.

With this background to the formal definition of a functional block, this can be developed to formally define virtualisation. This is set out in the four steps shown in figure 7.

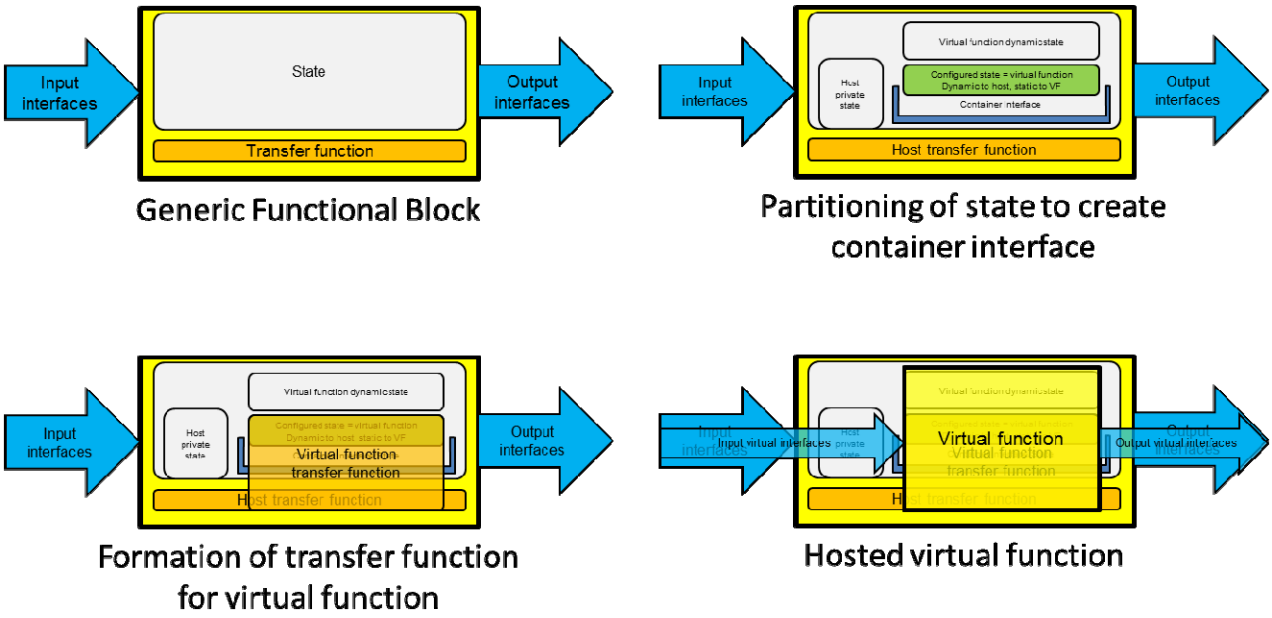

Figure 7: More detailed technical description of functional virtualisation

The first step shows a generic functional block with inputs, outputs, state, and a transfer function.

The second step shows the state of the functional block partitioned into three distinct types of state:

- **host private state** which holds the dynamic state which is private to the host function;
- **configured state** which will be configured and then held constant and invariant during the life of the virtualised function and which defines the virtualised function;
- **virtual function dynamic state** which will hold the dynamic state of the virtualised function.

This second step shows the essence of virtualisation: the fixing of some of the host's state so that it is held constant for the life of the virtualised function. The state which is available for fixing is the aspect of the container interface that allows the virtual function to be defined.

The third step shows how this fixed state combines with the host transfer function in order to define the transfer function of the virtualised function. This combination of the fixed state and the host transfer function allows the virtual function to execute.

The fourth step shows how the range of values defined for the host input and output interfaces are partitioned so that only those values which control the execution evolution of the virtual function are identified as belonging to the virtual interfaces of the virtual function. The input virtual interface is therefore defined as a subset of the range of values of the host function's input interface. Similarly, the output virtual interface is defined as a subset of the range of values of the host function's output interface. The behaviour of the virtual function as perceived through the virtual interfaces, is now exactly that of the virtual function.

With this formal definition of virtualisation, it demonstrates:

- a virtual function is configuration of it host function;
- a virtual function is an abstract view of the host function when appropriately configured.

6.1.5 Standardizing Organisations Impacting Virtualisation and Associated Interfaces

There does not appear to be many appropriate existing standards in this area and therefore an opportunity for new work. Appropriate bodies include:

- International Council on Systems Engineering (INCOSE).
- Object Management Group (OMG) work on Unified Modelling Language (UML) and related languages including SysML.

6.2 NFVI and Cloud Computing

The NFV white paper identified Cloud computing as an enabler for Network Function Virtualization ([i.11], page 9). The co-ordinated implementation of cloud and networking for enterprises, allowing on-demand services to be offered and providing capital efficiency for enterprise customers and network operators was also identified as a field of application for NFV ([i.11], page 7). Cloud computing supports a variety of different applications on a common infrastructure. The NFVI provides a common infrastructure that can support a variety of different fields of application as shown in figure 1.

Network Functions Virtualisation will leverage modern technologies such as those developed for cloud computing to deliver end-end network services using the NFVI. At the core of these cloud technologies are virtualisation mechanisms: virtualisation of hardware by means of hypervisors, as well as the usage of virtual Ethernet switches (e.g. vSwitch) for connecting traffic between virtual machines and physical interfaces. Cloud infrastructures provide methods to enhance resource availability and usage by means of orchestration and management mechanisms, applicable to the automatic instantiation of VNFs in the network, to the management of resources by assigning VNFs to the correct CPU core, memory and interfaces, to the re-initialization of failed VMs, to snapshot VM states and the migration of VMs. Finally, the availability of open APIs for management and data plane control, like OpenFlow, OpenStack, OpenNaaS or OGF's NSI, provide an additional degree of integration of Network Functions Virtualisation and cloud infrastructure ([i.11], pages 9 and 10).

The NIST definition of cloud computing [i.10] is used here as the basis of a discussion of NFV in terms of essential characteristics, service models and deployment models. Through this discussion, NFV can be seen as a type of Private Cloud IaaS where VNFs are executed in support of the services that a Network Operator provides. The execution environment for VNFs is provided by the NFVI deployed in various NFVI-PoPs. Centralization of management and data is posited as a benefit of cloud computing [i.17], but NFV delivers end-end services a geographically distributed NFVI.

NOTE: An International Standard/Recommendation on Cloud Computing overview and vocabulary covering similar aspects than provided in [i.10] is under preparation in ISO/IEC and ITU-T.

6.2.1 Essential Characteristics of Cloud Computing applied to the NFVI

The NIST definition of Cloud Computing [i.10] identifies five essential characteristics of cloud services: on-demand self- service, broad network access, resource pooling, rapid elasticity and measured service. Because NFV encompasses these five characteristics, it could be considered an application of cloud technology. In order to facilitate the reuse of tools, technologies and operational processes from Cloud Computing installations, the NFVI should provide equivalent support of these essential characteristics.

6.2.1.1 On-demand self-service in NFVI

The consumer of the NFV is most typically the Network Operator. The Network Operator expects to be able to unilaterally provision and allocate existing deployed NFVI resource capacity such as server time and network storage. In some cases the VNFs or the management and orchestration functions coordinating them may be the entities provisioning computing capabilities on behalf of the Network Operator.

6.2.1.2 Broad network access in NFVI

NIST describes broad network access in cloud computing as providing capabilities that are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms. NFVI-PoPs are expected to be accessed remotely (via the transmission and access network infrastructure). Virtualized network functions are expected to handle the network traffic of a variety of existing network elements and terminal types. The management and orchestration of virtualized network functions may be a type of virtualized network function as well. The network access capacity required at a particular NFVI-PoP will depend on the fields of application and the VNFs instantiated on the NFVI at that NFVI-PoP. Some VNFs may require location adjacent to specific types of network access facilities (e.g. the virtualised eNode B of use case #6 [i.3] and [i.4] would typically be located close to cellular towers). The NFV Architectural Framework [1] supports end-end network services delivered over a NFVI deployed in NFVI-PoPs as shown in figure 8.

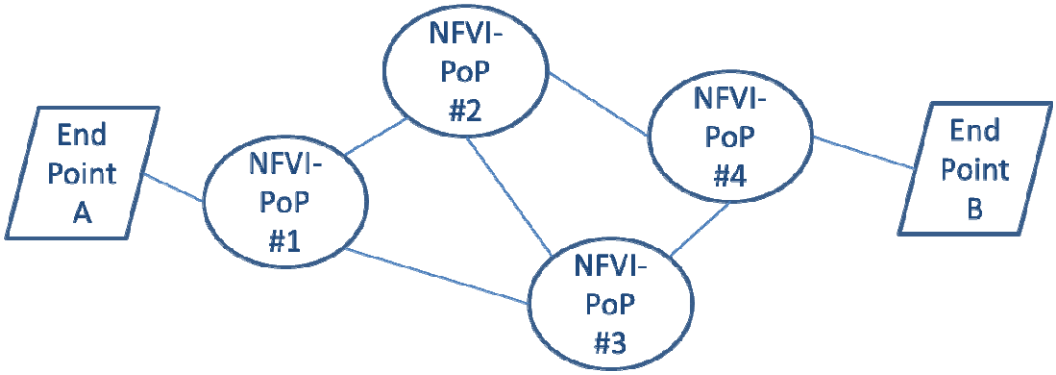

Figure 8: Example of Broad Network Access to NFVI

6.2.1.3 Resource pooling in NFVI

While a single VNF might be deployed at a single NFVI-PoP, in general, the NFVI supporting the VNFs is expected to typically service multiple VNFs from different VNF providers in a multi-tenant model at one NFVI-PoP. Resource demands on the NFVI are expected to change dynamically with the service load (whether data plane or control plane traffic) processed by the VNF. Location independence is an objective in that specific VNFs should not be constrained to only run on dedicated NFVI hardware resources; some VNFs may be location specific if required by their network function e.g. due to latency requirements, or the location specific nature of distributed network functions.

6.2.1.4 Rapid elasticity in NFVI

Network Operators normally require rapid elasticity with the automated provisioning and release of the computing, storage and communication resources of the NFVI allocated to specific VNF instances. Rapid scaling could be triggered in response to the service load processed by the VNF, failover and recovery mechanisms, management and orchestration driven rearrangement of resources for optimization of power, etc.

6.2.1.5 Measured service in NFVI

Network Operators expect that the management and orchestration of VNFs will normally require the automatic control and optimization of NFVI resource usage by the VNFs. This optimization implies that the use of those resources by individual VNFs is metered in some fashion appropriate to the type of NFVI resource (e.g. storage, compute, communication).

6.2.2 Service Models impacting the NFVI

NIST definition of Cloud Computing [i.10] describes three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Service Models in the context of NFV are discussed in both NFV Use Cases [i.3] and [i.4], and NFV Virtualisation Requirements [i.5].

6.2.2.1 SaaS

The VNFs are software applications running on the cloud like infrastructure. These applications provide network functions processing data and control plane traffic for the network operator rather than just web Services. The VNFs are expected to typically be executed according to a private cloud model by the network operator. Some categories of VNFCs may be amenable to be executed in a third party cloud infrastructure and utilized by the network operator in a SaaS model. Use Case #2 [i.3] and [i.4] provides a Service Model description for Virtual Network Functions as a Service (VNFaaS). VNFaaS delivers a network function remotely. Whether the deployment and operation of VNFaaS places any additional requirements on the NFVI beyond those of any other VNF are currently considered beyond to the scope of the present document.

6.2.2.2 PaaS

NFV provides an opportunity for the network operator to significantly improve the speed with which new services and applications are developed and deployed. The PaaS model provides a service model consistent with deployment by network operators of services constructed from configurations of multiple VNF instances. Use Case #3 [1] provides a Service Model description of a Virtual Network Platform as a Service (VNPaaS). VNPaaS provides to the user of the service a larger scope of control than that provided by VNFaaS. Whether the deployment and operation of VNPaaS places any additional requirements on the NFVI beyond those of any other VNF are currently considered beyond the scope of the present document.

6.2.2.3 IaaS

In IaaS, the capability provided is the provisioning of compute, storage and communication resources so that applications can be run on them. Here, the VNFs are the applications intended to be run and the virtualization infrastructure in which they execute provides an IaaS for the Network Operator. Use Case #1 [1] provides a Service model description of Network Function Virtualisation as a Service (NFVIaaS). NFVIaaS permits a Service provider to fulfil, assure and bill for services delivered to end users across NFVIs that are independently administered. The NFVI should provide the appropriate authentication mechanisms to ensure that only authorized entities have access to NFVI resources.

6.2.3 Cloud Deployment Models impact on NFVI

NIST definition of Cloud Computing [i.10] identifies four deployment models: Private Cloud, Community Cloud, Public Cloud, or Hybrid Cloud. Support for specific Cloud Deployment models is not identified in the NFV Virtualisation Requirements discussion on Deployment Models (see clause 6.2 of [4]).

This clause describes NFV in the terms of the existing NIST Cloud Computing deployment models in order to describe the fit of NFV with this model. In order to meet specific requirement of NFV, notably security requirements, NFV has a number of further constraints on deployment which go beyond these cloud deployment models. The NFV deployment models required for security are describing in clause 5.1 of ETSI GS NFV-SEC 001 [6] and summarized in table 1 of this document.

6.2.3.1 NFVI as a Private Cloud Infrastructure

In a Private Cloud, the cloud infrastructure is provisioned for exclusive use by a single organization. The NFVI is expected to be deployed by the Network Operator for its own use in providing network services.

6.2.3.2 NFVI Community Clouds

In Community clouds, the cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns. A specific community of consumers has not yet been identified. It is possible that the ecosystem of vendors and a network provider with an NFVI, or the set of Network Operators with appropriate agreements could be considered a community. Formation of Community Clouds is beyond the scope of the present document.

6.2.3.3 Public Cloud and NFVI

With Public Clouds, the cloud infrastructure is provisioned for open use by the general public. The NFVIaaS is not expected to be available to the general public. It is not expected that a Network Operator would deploy VNFs within a Public Cloud due to the locality, latency and throughput requirements of most network functions.

6.2.3.4 Hybrid Cloud and NFVI

Hybrid Clouds have an infrastructure that is a composition of two or more distinct cloud infrastructures. NFV could be operated on a hybrid cloud deployment model where a Network Operator maintained an IaaS private cloud for its own services, but offered on a wholesale basis, VNFaaS for enterprises, or other Network Operators.

6.2.4 Standardizing Organisations for Essential Characteristics of Cloud Computing applied to the NFVI

Relevant standards bodies include:

- **NIST** has a number work items on cloud standardsincluding definition the services and deployment models outlined above [i.10].
- **Open Data Centre Alliance (ODCA).** The ODCA is defining usage models for cloud computing.
- **ITU-T SG13 [i.21]** is working on questions covering Cloud, SDN, and NFV, is hosting the ITU-T Joint Coordination Activity on Cloud (JCA-Cloud) and SDN (JCA-SDN).
- **ISO/IEC JTC1 SC 38 [i.20]** has a number of work items on cloud standard including Cloud Overview and Vocabulary and Reference Architecture.
- **Open Networking Foundation (ONF)** is developing architecture and interface standards of Software Defined Networks (SDN).
- **OpenStack** is an open source projects developing a cloud management system open source implementation.
- **Open Grid Forum** is creating standards for open distributed grid computing.

6.3 Domains and Inter-Domain Interfaces

At every phase of engineering of telecommunications, whether service architecture and design, network architecture and design, operations of service delivery, or operations of service maintenance, it is characterized by very high levels of complexity.

NOTE: Complexity is used here in fairly general terms and it not intended to the specifically associated with any particular normal definition or measurement method.

This complexity arises from a number of necessary features of telecommunications including:

- the very large number of interconnected physical components necessary to provide global, universal connectivity;
- the growing range of services demanded of telecommunications;
- the pace of technology change creating heterogeneity in the technology base;

- the need to offer continuous, uninterrupted service (the global telecommunications network has been in continuous, uninterrupted operation of over 100 years);
- the need to interconnect many commercially, legally, and politically autonomous organisations.

Over the years telecommunications operators and vendors have founds ways of containing this complexity to achieve engineering solutions that are manageable, scalable, and evolvable.

One of the primary motivations and benefits of NFV is that it can improve the management of complexity through the separation of software from hardware. These can be separately developed and evolved greatly reducing the dependency between the two that is one of the many sources of complexity.

Lack of scalability is potential source of complexity and cost. This arises when there is a non-linear cost to the size of a 'unit' of resource and it is cost effective to have many smaller units of resource rather than a few larger units of resource. When this is the case, a complexity grows in the way in which all the smaller units should be interconnected and used in order to meet the overall requirements.

NFV will necessarily change some of the optimal size of resource units and will change some of the way in which efficient scaling can be achieved. Moreover, there is now a new relationship between units of software and units of hardware, which also create opportunity to improve the scalability of overall solutions.

The architecture of NFV generally and the NFV infrastructure specifically are defined and set out in a way to manage both complexity and scalability.

Two specific techniques used to manage complexity and scalability of the infrastructure are:

- partitioning of NFV and its infrastructure into largely separable autonomous domains;
- abstraction of systems and their interfaces (described in ETSI GS NFV-INF 007 [i.15]).

As part of the primary overall objective, NFV separates domains associated with VNFs from domains associated with the NFVI.

However, even within the NFVI, there is considerable merit in identifying specific domains with the NFVI.

It is already the case that the purchase of servers with storage, network switches, and hypervisors are frequently decoupled. It is possible to conclude from this observation of the current market that there are already sufficient standards to support interworking between these functional domains. Most network switches from most network switch vendors will interconnect which most servers from most server vendors. Similarly, most hypervisors from the different hypervisor vendors will run on most servers from most server vendors. Such practical levels of interoperability require that the interfaces be both reasonably well understood and reasonably enduring.

The NFVI architecture fully supports these broad domains and fully supports the continued open supply between these domains. The NFVI is split into three primary domains.

- The compute domain (including storage) - which comprises the generic high volume servers and storage.
- The hypervisor domain -which comprises the hypervisor and which:
  - provides sufficient abstraction of the hardware to provide portability of software applications between different servers (this is potentially less significant for NFV that it is for public cloud);
  - allocates the compute domain resources to the VMs;
  - provides a management interface to the orchestration and management system which allows for the loading and monitoring of VMs.
- The infrastructure network domain - which comprises all the generic high volume switches interconnected into a network which can be configured to supply infrastructure network services.

It is also fully understood that the definition of these domains are neither precise nor orthogonal (ie there may be functional overlap). This possibility of some functional overlap between domains is accommodated in the architecture.

In addition, closely associated with these three NFVI domains are the NFV Management and Orchestration entities that are expected to run on the infrastructure as a set of modular, interconnected virtual machines. These entities perform all the automated orchestration and management operations needed to load and manage software appliances running on the infrastructure.

Figure 9 illustrates the application of the principle of domains to NFV and in particular to the NFVI. A number of points arise from this application of the principle of domains which are illustrated in this figure:

- The architecture of the VNFs is separated from the architecture hosting the VNFs, that is, the NFVI.
- The architecture of the VNFs may be divided into a number of domains with consequence for the NFVI and vice versa.
- Given the current technology and industrial structure, compute (including storage), hypervisors, and infrastructure networking are already largely separate domains and are maintained as separate domains with the NFVI.
- Management and orchestration tends to be sufficiently distinct from the NFVI as to warrant being defined as its own domain, however, the boundary between the two is often only loosely defined with functions such as 'element management functions' in an area of overlap.
- The interface between the VNF domains and the NFVI is a container interface and not a functional block interface (this is emphasized in this figure).
- The Management and Orchestration functions are also likely to be hosted in the NFVI (as virtual machines) and therefore also likely to sit on a container interface.

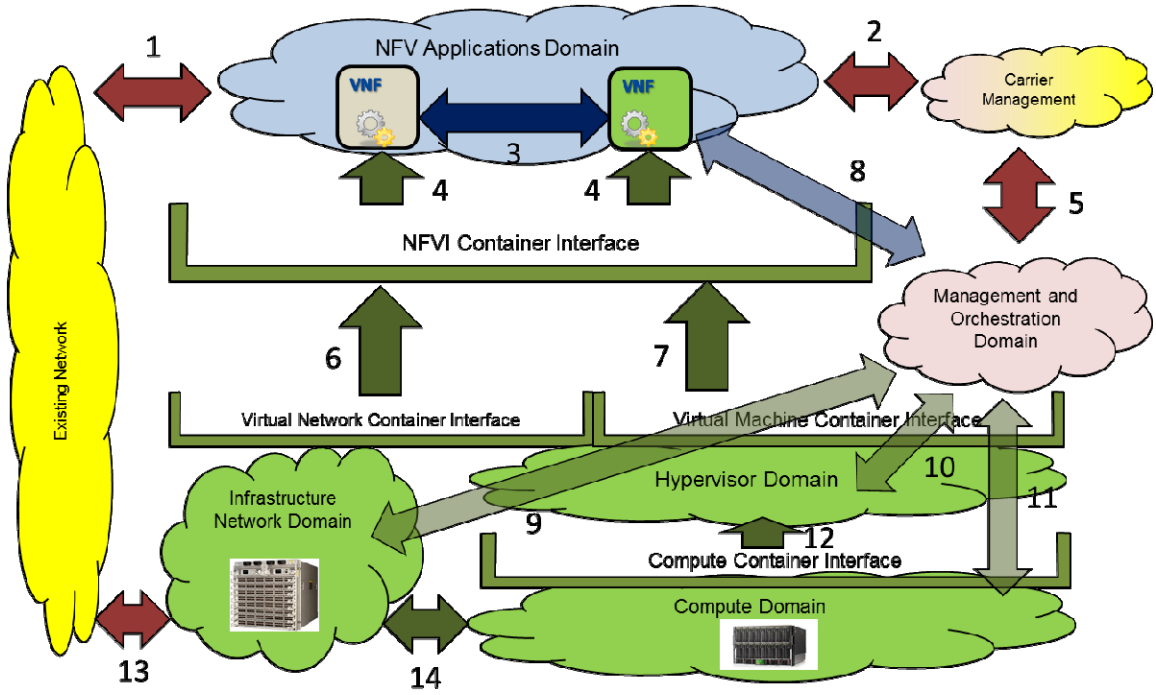

Figure 9: General Domain Architecture and Associated Interfaces

Figure 9 also sets out the primary interfaces. Table 1 catalogues the interfaces identified in figure 9.

Table 1: Catalogue of Inter-domain Interfaces Arising from Domain Architecture

Interface Type	#	Description
NFVI Container Interfaces	4	This is the primary interface provided by the infrastructure to host VNFs. The applications may be distributed and the infrastructure provides virtual connectivity which interconnects the distributed components of an application.
VNF Interconnect Interfaces	3	These are the interfaces between VNFs. The specification of these interfaces does not include, and is transparent to, the way the infrastructure provides the connectivity service between the hosted functional blocks, however distributed.
VNF Management and Orchestration Interface	8	The interface that allows the VNFs to request different resources of the infrastructure, for example, request new infrastructure connectivity services, allocate more compute resources, or activate/deactivate other virtual machine components of the application.
Infrastructure Container Interfaces	6	**Virtual Network Container Interface:** the interface to the connectivity services, for example E-Line and E-LAN service, provided by the infrastructure. This container interface makes the infrastructure appear to the NFV applications as instances of these connectivity services.
	7	**Virtual Machine Container Interface:** the primary hosting interface on which the VNF virtual machines run. NOTE: NFV orchestration and management functions are expected to run on this interface.
	12	The primary compute hosting interface on which the hypervisor runs.
Infrastructure Interconnect Interfaces	9	Management and Orchestration interface with the infrastructure network domain.
	10	Management and Orchestration interface with the hypervisor domain.
	11	Management and Orchestration with the compute domain.
	14	Network interconnect between the compute equipments and the infrastructure network equipments.
Legacy Interconnect Interfaces	1	The interface between the VNF and the existing network. This is likely to be higher layers of protocol only as all protocols provided by the infrastructure are transparent to the VNFs.
	2	Management of VNFs by existing management systems.
	5	Management of NFV infrastructure by existing management systems.
	13	The interface between the infrastructure network and the existing network. This is likely to be lower layers of protocol only as all protocols provided by VNFs are transparent to the infrastructure.

A number of points which characterize the NFV architecture emerge from this list of interfaces:

- A VNF can be decomposed and made up from sub-parts which are themselves VNF which are interconnected by the infrastructure.
- VNFs can autonomously request orchestration and management changes, for example the instantiation on new VNFCs, the deletion of a VNFC, the changing of the allocation of resource to a VNFC, etc.
- Network interfaces shall be decomposed between the infrastructure protocols and the VNF protocols such that one set is fully transparent to and independent of the other.
- The infrastructure network (and other networking functionality in the compute and hypervisor domain such as NICs and vswitches respectively) is completely abstracted by the network infrastructure container interface.
- The non-networking aspects of the compute domain are abstracted as far as is appropriate by the compute container interface.
- The NFV infrastructure, applications with its orchestration and management is fully interoperable with existing carrier networks and carrier OSS/BSS systems.

6.3.1 Standardizing Organisations Impacting Domains and Inter-Domain Interfaces

NFV and its architecture are new. It is noted that Recommendation ITU-T SG13 [i.21] has a number of questions working in this area.

6.4 Multiplicity, Composition, and Decomposition

This clause provides an overview of the way functional block methodology is developed to describe, define, and specify the mapping of VFBs to HFBs. Details of the way functional block methodology is used to describe, define, and specify this mapping are set out in "Methodology to describe: Interfaces and Abstractions" (ETSI GS NFV-INF 007 [i.15]).

6.4.1 Principles of Multiplicity

The principles of virtualisation outlined and defined in clause 6.1 describe an elementary case of virtualisation where one host function hosts one VNF. However, virtualisation is considerably more flexible than this.

First, one host function, for example a compute node which is defined more formally below, can host more than one virtual function. Figure 10 shows the important example of a compute node which hosts a hypervisor. The hypervisor is itself able to host many virtual machines (VMs) each of which can host a VNF as illustrated in the figure. The single VNF hosted directly on a single VM (a one to one mapping) is called a VNF Component (VNFC).

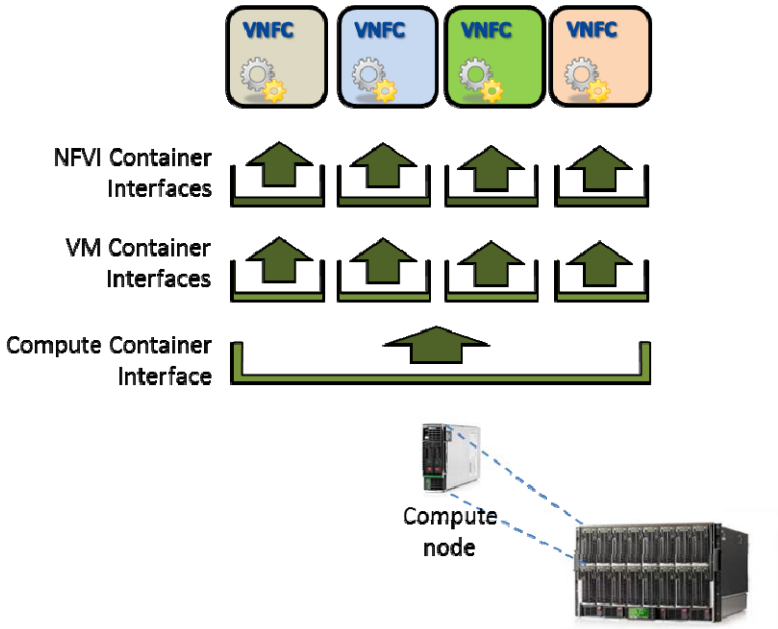

Figure 10: A single compute platform supporting a multiplicity of VNFCs

As important is the recognition that the NFVI is a distributed system and hosted VNFs may also be distributed systems. The NFVI comprises compute nodes and infrastructure networks. The hypervisors hosted on the compute nodes provide virtual machine container interfaces while the infrastructure network provides infrastructure connectivity container interfaces. These infrastructure connectivity container interfaces provides connectivity services such as E-Line and E-LAN services as defined by the MEF. These services are virtual functions hosted on the infrastructure network.

When put together, these VM container interfaces and virtual network container interfaces provide a composite NFVI container interface which is distributed. This distributed NFVI container interface can host distributed VNFs. This is illustrated in figure 11. The figure shows a composite, distributed VNF hosted on the composite, distributed NFVI container interface. It also shows the constituent VNFCs hosted on the constituent VM container interfaces and the virtual interfaces of the VNFCs hosted on virtual network container interfaces.

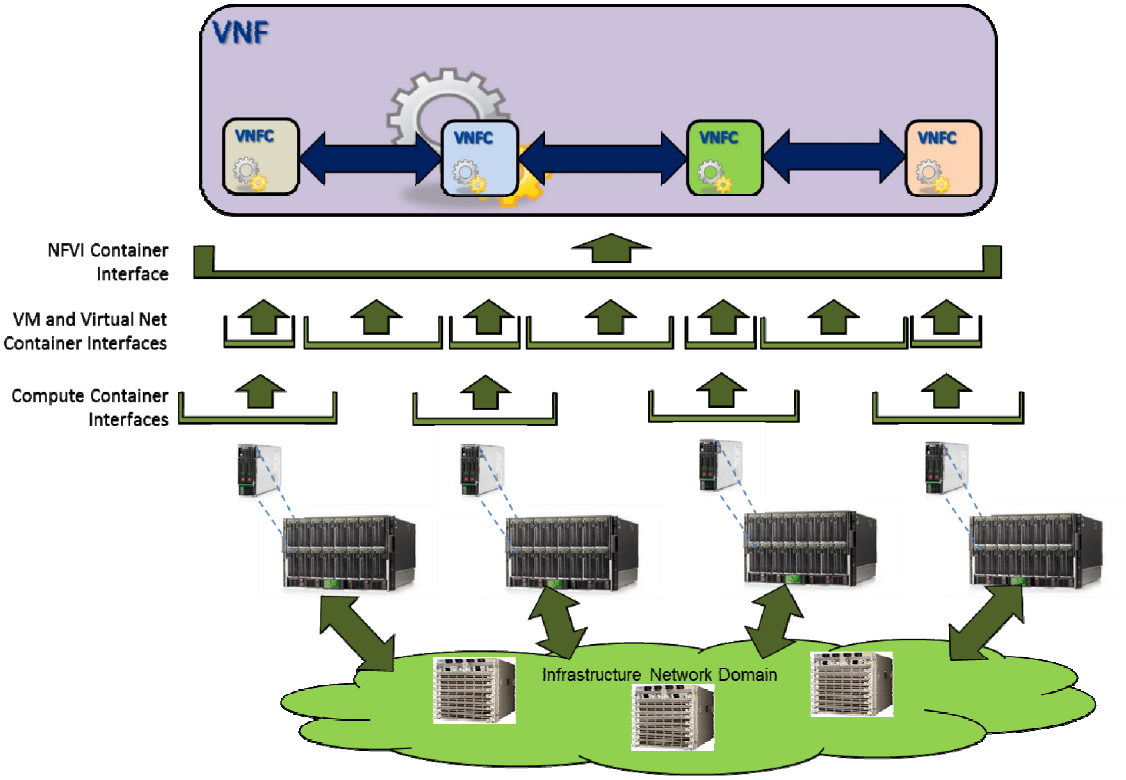

Figure 11: A composed, distributed VNF hosted across a multiplicity of compute platforms

This architecture can be more generally layered recursively. NFVI container interfaces can be composed of, or decomposed into, a set of NFVI container interfaces and virtual network container interfaces.

There are important aspects of distribution which affect the execution performance of VNFs which places constraints on this recursive architecture. There is a practical importance to defining a compute node where the speed of synchronous execution is not impaired by distribution. This is described in more detail in the clause on the compute domain below apply for network functions.

6.4.2 NFVI Implications of Complete and Partial Virtualization of Network Functions

Network Operators and Service Providers deliver end-end network services. Figure 12 shows an example network service delivered between End Point A and End point B. In this example, the end-end network service requires processing by three Network Functions {Nfa, NFb, NFc} that are implemented in three Network Elements {NE1,NE2,NE3 respectively}. In the general case, the three network elements could be located in three different N-PoPs {N-PoP1, N-PoP2, N-PoP3}.

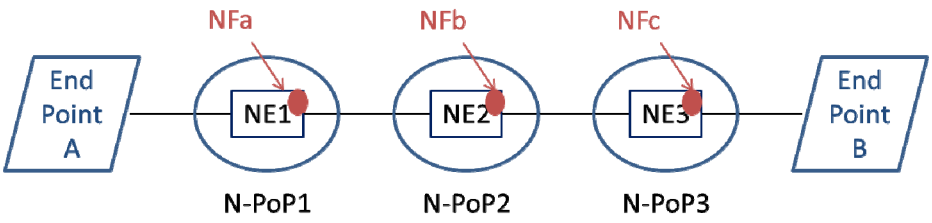

Figure 12: Example End-End Network Service

When considering the virtualisation of Network Functions (e.g. Nfa, NFb, NFc) then a number of operational considerations apply in different cases depending upon:

- the degree of virtualization (complete or partial);

- whether the VNFs are from a single or multiple vendors;
- whether the VNFs can be deployed in parallel across multiple VMs;
- whether the VNFs have state;
- and whether they are deployed across a single or multiple operators.

6.4.2.1 Complete and Partial Virtualization

A large scale Network Function (NF) or a Network Element may be decomposed in a number of constituent NFs. If *all* the constituent NFs are implemented as VNFs, then the virtualisation of the large scale NF or Network Element can be said to be complete. If some but not all of the constituent NFs are implemented as VNFs, then the virtualisation of the large scale NF is said to be partial. However, whether the decomposition is complete or partial, the result is likely to be multiple VNFs. However, a consequence of virtualisation is that while non-virtualised constituent NFs will be implemented at fixed network location, the VNFs may be implemented at any location in the NFVI, subject to hosting policy constraints.

Network functions operating at or above Layer 3 (e.g. firewalls, load balancers, etc.) may be representative of the types of Network functions that may be completely virtualized.

6.4.2.2 Decomposition of VNFs and Relationships between VNFs

Virtualization of a NF implies that the equivalent VNF accurately emulates the behaviours of that NF when executing on specific NFVI. A large scale VNF may be composed of one or more constituent VNFs connected together in some form of a forwarding graph, a characteristic common to the NFV use cases under consideration.

NOTE: A VNF may be composed recursively to define yet another composition; for example a virtual Evolved Packet Core (vEPC) VNF is made up of other lower-level VNF instances of (at least) an S-GW, MME and P-GW.

For the purposes of this discussion, each constituent NF of a large scale NF is identified as a 'functional node' in the forwarding graph and would normally be implemented as a single logical device when in a non-virtualised form.

The cases of interest are where constituent VNFs interface to other constituent VNFs as well as to any non-virtualised constituent NFs. An individual constituent VNF can have the following deployment cases:

- a 1:1 implementation of single Network Element's Network Function by a single VNF;
- an N:1 case where there are N parallel constituent VNFs implementing the capacity of a single Network Element's Network Function;
- a 1:N case where N Network Elements's Network Functions are implemented by a single VNF (for example, a virtualized home gateway).

6.4.2.3 1:1 VNF Implementation of a Network Element by a VNF

If there is a 1:1 mapping case of a VNF to a Network Element, then external interfaces and management will likely be completely specified by the existing Network Element standardized interfaces. The 1:1 mapping, especially when used complete virtualization, reduces the interoperability challenge to largely one of conformance with the standardized interfaces of the Network Element. Figure 13 illustrates this 1:1 mapping where Nfa was implemented in NE1 deployed at N-PoP1, and supports interfaces i_1 to i_n. In this example NE1 provides no network functions other than Nfa. The equivalent Virtualised Network Function (VNFa) executes in the NFVI Node and provides the same interfaces i_1 to i_n. If the NFVI Node implements no other VNFs, then it could be considered a direct replacement for NE1. For a direct replacement of an NE by an NFVI node running the equivalent VNFs at the same location, NFVI-PoP1 would be the same location as N-PoP1.

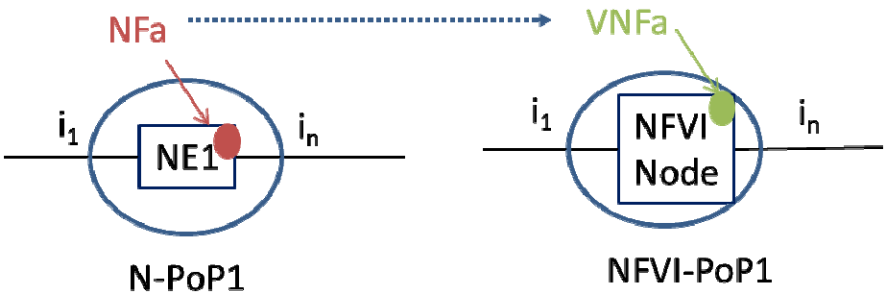

Figure 13: Mapping NE:VNF 1:1

Figure 14 shows the case where the end-end Network Service of figure 12 is implemented using 1:1 mapped VNFs.

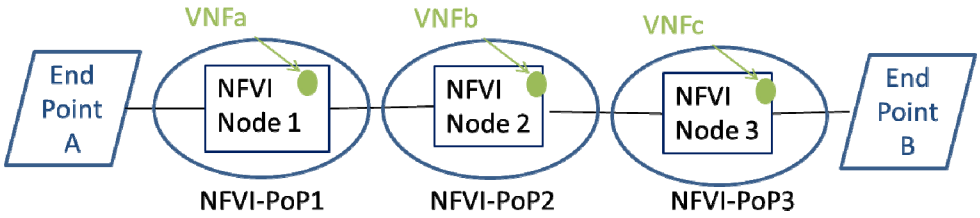

Figure 14: Example End-End Network Service using 1:1 mapped VNFs

6.4.2.4 N:1 Implementation of a Network Element by Parallel VFNCs

It is possible that a constituent VNF may itself be decomposed such that it is composed of a set of parallel VNFCs. This may be done within a constituent VNF by vendor implementation to improve efficiency, scaling, and/or performance.

If a constituent VNF has interfaces to neighbouring constituent VNFs and a deployment has these constituent VNFs from different vendors, then these interfaces of the constituent VNFs shall be standardized. As part of their operation, the standards shall ensure that the common state (for example, subscriber and service state) is maintained, and that management functions are provided in accordance with the general specification of the implemented constituent NF. For example, when parallel (N:1) VNFCs are implemented at a standard interface, functionality is needed to distribute traffic between these parallel VNFCs. The functionality which distributes traffic shall take account any shared state. For example, if the VNFC is session aware, all packets of each session will be sent to the same VNFC.For example if two functional nodes share the state of a subscriber, then traffic related to that subscriber shall be sent from a VNFC holding that subscribers state to the next VNFC holding state for that subscriber.

One particular example of decomposing an NF into two constituent NFs would be the separation of a routing function into independent components for the control plane and data plane using protocols such as Openflow or FORCES. More specifically, a service edge router supporting multiple services could be decomposed into a single non-virtualized data plane NF and multiple independent virtualized control plane functions for the control planes: one for each service type. Another data plane example would be to decompose a firewall NF into multiple parallel VNFCs, each holding the state of multiple potential firewall customers.

Some constituent VNFCs could be identified as "replicable" within a configured bound on minimum/maximum resources. The number of parallel instances for these replicable VNFCs could either be increased/decreased based upon performance triggers (e.g. CPU, virtual memory paging, etc.). This may be achieved, for example:

1) automatically by MANO based on indication in the associated VNFC configuration descriptor; or

2) based on run-time request to MANO from a VNF instance.

State consistency would be maintained by this set of replicable VNFCs. Figure 15 illustrates the case of the N:1 mapping of the Nfa implemented in a single high capacity Network element (NE1) into three instances of VNFa. The instances of VNFa, in general, may be executing in different NFVI Nodes in different NFVI-PoPs, and in this figure they are shown executing in two different NFVI Nodes in two different NFVI-PoPs. In aggregate the combination of NFVI Nodes 1 and 2 and the split/merge functions prove the equivalent set of external interfaces i_1 to i_n. In this example the splitting and merging functions are allocating traffic across instances of the same type of VNF (VNFa). This sort of splitting and merging function is typically referred to as load balancing. Load balancing is identified in Use Cases #4 (VNF Forwarding Graph), #6 (Mobile Base Station) and #7 (virtualised home environment) of [1].

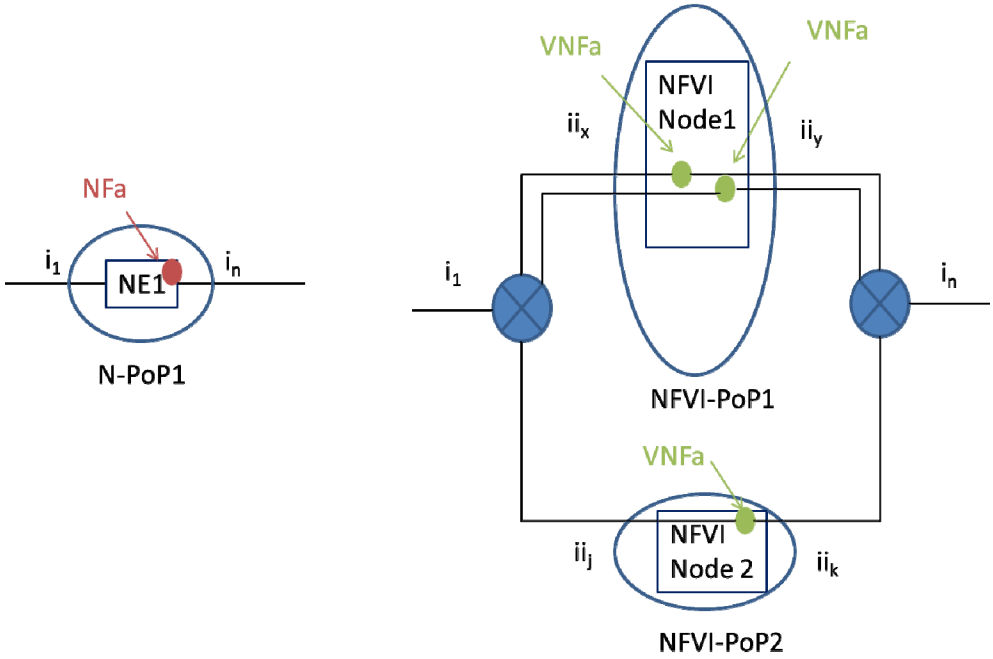

Figure 15: N:1 Mapping of VNF (NFVI-Node): NE

6.4.2.5 1:N Multiplexed Implementation of Multiple Network Elements by a Single VNF

The 1:N case is where N Network Functions are implemented by a single VNF (for example, a virtualized home gateway) and where each individual NF is defined by at least is individual state. Figure 16 provides an example of three identical NFs implemented in different Nes in different locations. The equivalent VNF (VNFa) supports a larger number of interfaces (i_j to i_k) from a single instance. Use case #7 (virtualised Home Environment) of [1] provides some discussions of this modularity. The single VNF will operate some means of partitioning between the individual NFs, but in this scenario, this level of partitioning would normally be vendor implementation specific.

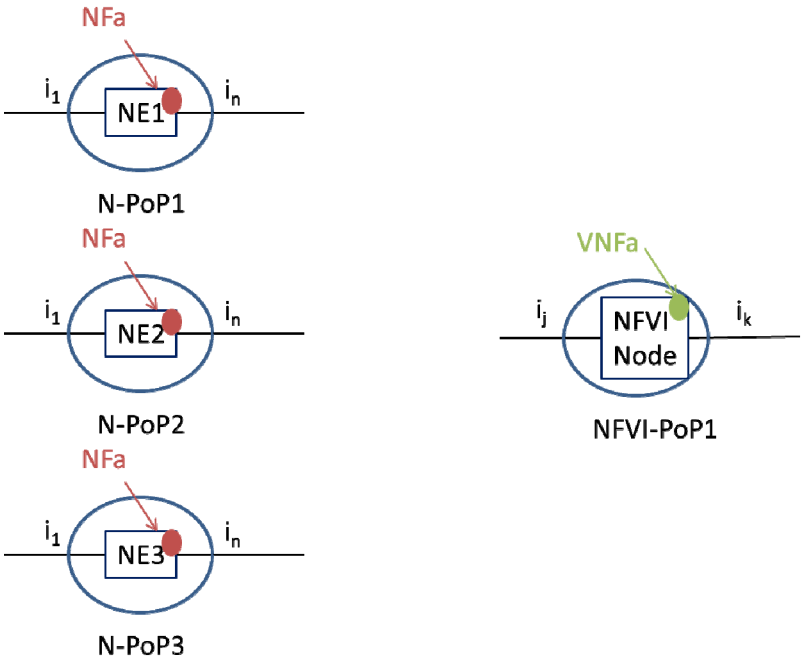

Figure 16: Example 1:N Mapping of VNFs:Nes

6.4.2.6 Shared Virtual Network Function Component Instances

Additionally, across a large scale VNF forwarding graph, there may be some constituent VNFs that are shared across more than one large scale VNF. These shared constituent VNF may be deployed at the VM level or may be deployed at a higher level of composition. Shared constituent VNF may create additional security, performance and management challenges. Most likely, shared constituent VNF would be effectively achieved within a closed vendor environment. If this is to be achieved in an open vendor environment, this is more likely to achieved when a vendor implementation of a large scale VNF, for example a vEPC, is composed from constituent VNFs, for examples a vS-GW or vP-GW which are themselves vendor interoperable. Making such shared VNF components supportable by multiple vendors will have issues similar to the parallel (N:1) and multiplexed (1:N) cases.

6.4.2.7 Relationship of Virtual Network Functions to Orchestration

Any parts of an NF that are virtualized as constituent VNFs should be described by a VNF Descriptor (VNFD) that includes information on the NF capacity, physical/logical interface connectivity, (pointers) to executable images and configuration databases, processing/networking/storage access capacity and performance requirements as input to NFV Orchestration (NFVO). NFVO would then take the VFND and map the VNF components to compute/networking/storage host functions and network topology configurations needed to realize the NF.

6.4.2.8 Other Aspects of Virtual Network Function Decomposition

Partial virtualization, as defined above, has impact onto the network management systems because the devices to be managed are not necessarily 1:1 replacements. The hosting of a VNFC on a specific compute node at a specific location has operational impacts in terms of reliability as well as fault detection, performance, security and operational procedures for the administration of the overall NF. This makes controlling these parameters for the overall partially virtualised NF more challenging as the overall NFs shall still conform to standard interfaces and present interoperability in data, control and management planes.

Partial virtualization may create additional interoperability challenges associated with the parallel (1:N) scenario, for example, if the separation of control and data planes are provided by different vendors. Generally, the level of decomposition potentially creates greater interoperability challenges because of the increased number of constituent parts.

SDN advocates the separation of the control and data functions of a Network Element. The data plane functions remain in a residual Network Element. The control functions are implemented in software and may be executed on an NFVI node. The interface IF1 between the SDN Controller (SDN-C) and the residual NE typically uses a protocol such as OpenFlow. The Open Daylight Project provides an open source implementation of SDN controller software in java. Figure 17 provides an illustration of this form of partial virtualisation where the SDN-C is executing in the NFVI Node. The functional mapping between the SDN-C functions and the functions identified in the NFV Architectural Framework [2] is for further study.

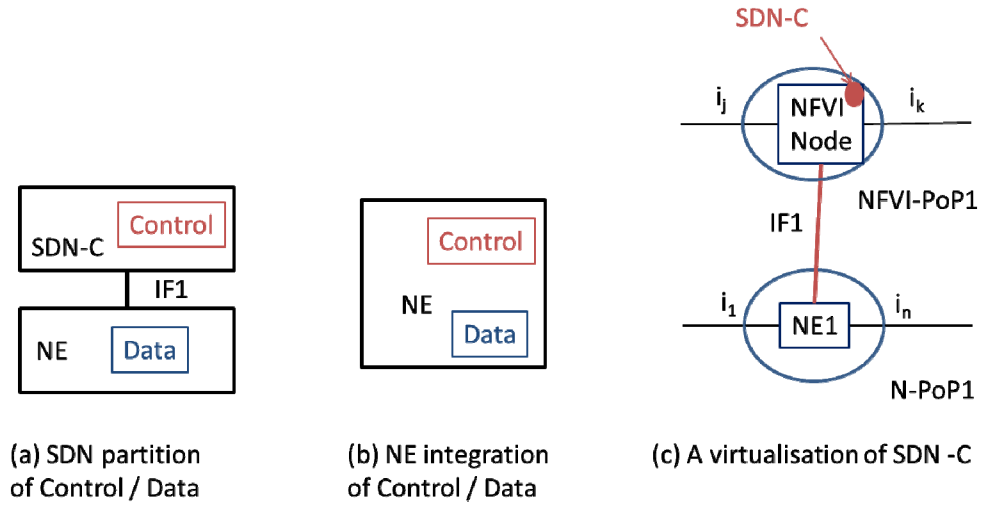

Figure 17: Partial Virtualisation using SDN to separate control and data functions

Most of the NFV use cases virtualize Network Functions within the administrative control of a single operator. This reduces the scale of the multivendor interoperability problem faced by an operator to the set of vendors selected by that operator. Inter-operator virtualization is feasible, but differences in management approach, dimensioning of resources, performance monitoring/reporting and resiliency may also need to be addressed.

However, virtualization of network functions opens opportunities for the creation of services that aren't modelled by current non-virtualised solutions or by SDOs, and the NFV might then suggest development of new standard interfaces, state definitions, forwarding graph particulars and management aspects to the relevant SDOs.

6.4.3 Standardizing Organisations Impacting Multiplicity, Composition, and Decomposition

NFV and its architecture are new. Both NIST and ONF activities are relevant and likely to be applicable to this area. ITU-T and others are initiating work in this area. The OpenDaylight implementation of an SDN controller may also be relevant.

6.5 Economics and Practical Interoperability

6.5.1 Interoperability and Hierarchical Interfaces

Interoperability is often held to be one of the primary motivations behind standardization. Broadly, if an interface is interoperable, it means that a component on one side of an interface will successfully interact with a component on the other side of the interface. In economic terms, this should imply that it should be possible to replace one implementation of a component with another implementation.

The telecommunications context of interoperability normally is that of peer level interfaces, for example, transport interfaces, signalling interfaces, and routing protocol interfaces. In this peer level context, each side of the interface is broadly equivalent. The ability to replace the implementation of a component on one side of the interface is equivalent to the ability to replace the implementation on the other side of the interface.

However, this is not the case with client-server interfaces such as those frequently in web services and data centric applications. As the name suggests, these interfaces are not peer to peer. While the primary definition of the client server follows from the operation of the interface, it always indicates differences in replicability on the two sides of such interfaces.

As the 'service' is defined by the server, the specification of the interface is bias to the server. In practice, it is often the case that a server which 'publishes' its interface and the server specification is likely to pre-exist clients. As a result, clients often have to adapt to the interface specification published for the server.

This hierarchy in the interface is even more marked with programmatic interfaces such as APIs, and especially with the container interfaces of the NFVI. The specification of the container interface of the NFVI will generally pre-exist the VNFs which it hosts. In general the VNFs need to be built to conform to the NFVI container interface specification rather the NFVI built to conform to the specific needs of a VNF.

With hierarchical interfaces, interoperability is open to wider interpretation. This can be seen by considering three illustrative levels of interoperability:

- **Freedom to use:** A vendor publishes a specification which can allow anyone to interwork with the client end of the interface. However, the specification itself is owned by the vendor and implementation of the server end is closed and proprietary.
- **Freedom to implement:** A body publishes a specification which allows for the open implementation of both the client end and the server end. However, the specification itself is owned by the body and others are not free to propose evolutionary changes to the specification.
- **Freedom to change:** As well as allowing freedom to use and freedom to implement, the specification is maintained by an open participation standards body which accepts reasonable contributions for evolutionary change to the specification.

While the general history of telecommunications standards is towards the 'freedom to change' class of interoperability, this is not the general case with IT standards, including those within OSS and BSS. Indeed, in this case there are many interfaces which are often described as interoperable which are in the first category or 'freedom to use', including some cloud management interfaces relevant to NVF.

The NFVI architecture aims to be as open as possible. However, it also aims to be practical about where openness is useful to the overall aims of NVF. The architecture, depending on the context of a practical interface, may accept any of the above definitions of interoperability. However, in other cases, the architecture may require full 'freedom to change' interface specifications.

6.5.2 Economics and Interoperability

As well as including a wider spectrum of definitions of interoperability especially for hierarchical interfaces such as client server interfaces, APIs, and container interfaces, NFV requires a wider spectrum to the defining interoperable conformance.

While it is sometimes useful to consider conformance as a binary selection - an implementation meets a required specification or it does not, this is often unhelpful in a broader economic analysis. It is rarely the case that any implementation is exactly to a specification, even if the specification is itself precisely defined. Generally, the more practical question in how close the implementation is to the specification, especially when taken in its overall system context. This question, does not normally give a simple binary answer, but often a more complex and context dependent answer.

In addition, even when an implementation does not meet a specification, it is rarely the case that effective interoperability cannot be achieved. Practical interoperability is normally achieved by creating an interworking function which able to translate between the implementations on either side of the interface. In this case, the degree of conformance to a specification can be assessed by the complexity of the interworking function required to achieve interoperability. These interworking functions, especially when taken in the context of hierarchical interfaces, are often referred to as 'stubs'.

This leads to the general test of interoperability used within the Infrastructure Architecture - the test of the interoperability of an interface is measured by the complexity of any interworking function needed to achieve effective interoperability.

Importantly, this test can also serve as the basis of an economic test of the cost of interoperability which can be used to set against the value of the interoperability.

Using this measure of interoperability, and taking into account that interfaces may be hierarchical and not peer to peer, it is possible to have four different scenarios:

- Scenario 1 - there is a prior agreed common interface standard which all parties implement and use.
- Scenario 2 - each higher level component (e.g. client) declares its interface and the lower level components (e.g. servers) each implement a customized 'stub' to interpret their internal system to the declared interface of the higher level components.
- Scenario 3 - each lower level component (e.g. server) declares its interface and the higher level components (e.g. clients) each implement a customized 'stub' to interpret their internal system to the declared interface of the lower level components.
- Scenario 4 - a fresh interface specification is developed for each pairing of lower level and higher level components.

The interworking functions associated with these scenarios are illustrated in figure 18.

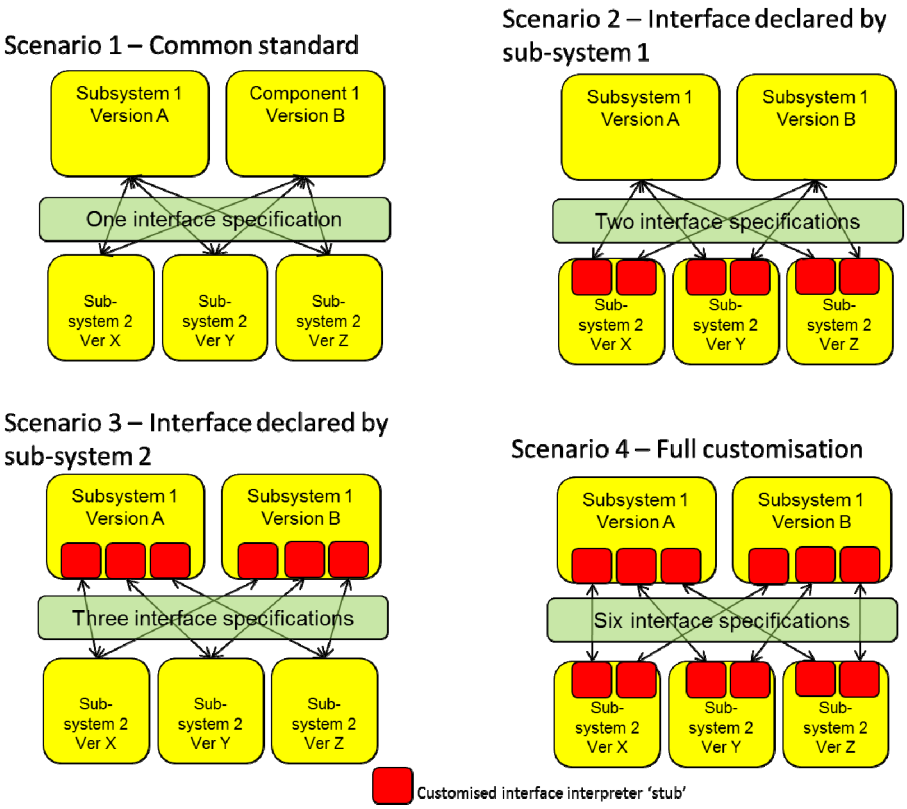

Figure 18: The use of translation 'stubs' to facilitate interworking

Using this framework, it is possible to address the following questions, which are central to the objective of the NFV, in both meaningful technical terms and economic terms.

First, looking at the NFVI from, the perspective of the VNFs.

- How can a VNF be ported from one (set of) locations of NFVI to a different (set of) locations of the same NFVI?
- How can a VNF be ported from one instance of a specific NFVI implementation to another instance of the same specific NFVI implementation?

EXAMPLE: A specific instance could be a set of specific releases from a specific set of vendors.

- How can a VNF be ported from one instance of a specific NFVI implementation to an instance of a different NFVI implementation?

Second, looking at the VNFs from the perspective of the NFVI.

- Can a provider of NFVI change of the implementation of a component within the NFVI without affecting the operation of any VNF?

6.5.3 Economic Analysis of Network Function Virtualization

The identification of appropriate methods for economic analysis of cloud computing applications is still evolving [i.3] and [i.4]. Methods for economic analysis of virtualization of network functions are even more nascent, with NFV business models, commercial ecosystems and market structures still in development. The development of standardized methods and economic models for NFV is not an objective of the present document. The discussion in this clause is provided as an illustration of one, and by no means the only, approach to illustrate the economic benefits of NFV. Common industry accounting mechanisms typically categorize costs separately for obtaining, installing and maintaining components, whether software or hardware. With existing network functions, the custom hardware and software components are typically deployed in fixed ratios, and often have significant lags in deployments of additional capacity because of their custom configurations. In contrast, multiple virtualized network functions may be dynamically deployed or reconfigured on COTS hardware. Capacity upgrades of COTS hardware have significantly less lags in deployment. Figure 19 illustrates the increased flexibility from virtualization. Evolving business models, commercial ecosystems and market structures for NFV may well develop other economic structures different to this example.

An important advantage of NFV for service providers is the ability to develop new network services through sequencing and orchestration of VNFs into VNF Forwarding Graphs that make available instances of new types network service to end users. This ability to 'mashup', that is experiment and test, new types of network services and rapidly validate the functionality of these services using the same NFVI as would be used for deployment provides important flexibility for the operators. This development and operational flexibility permits the operators to significantly increase the velocity of new service development.

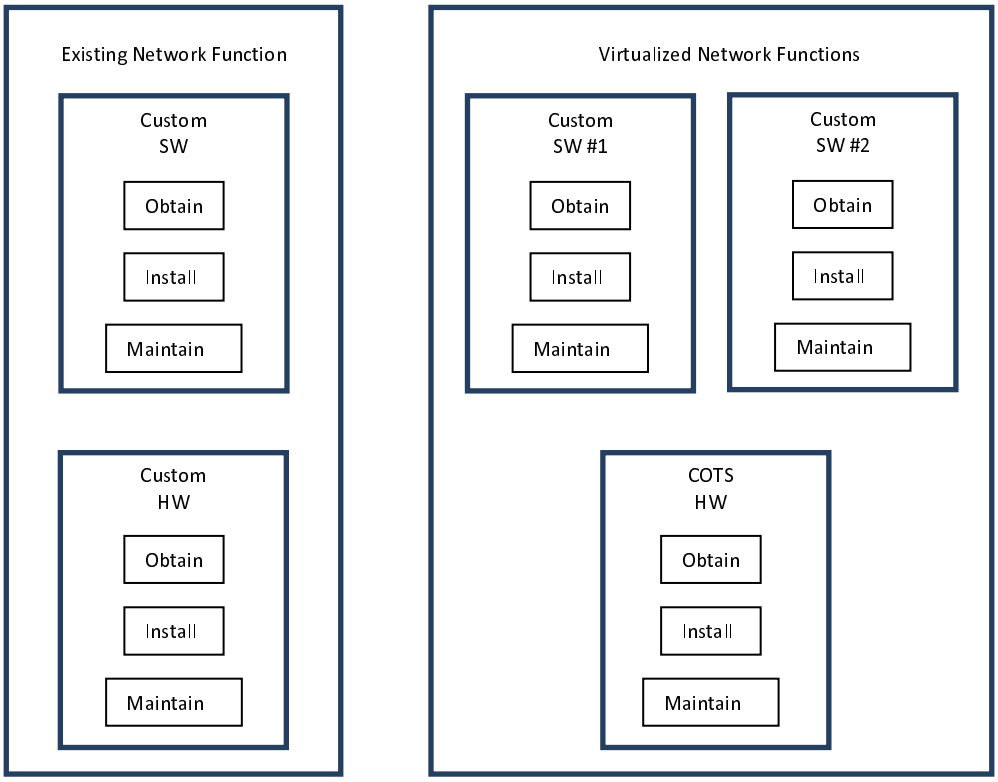

Figure 19: Illustrative Example for Economic Analysis of Virtualization

6.6 Key Quality Indicators for the NFVI

In order to insure that it is feasible and likely that VNFs deployed on the NFV infrastructure can deliver a consistent and acceptable service quality to end users, as well as rapidly isolate and correct failure conditions, robust quantitative measures are required for key quality indicators of the NFVI services. These quantitative measures need to reflect the way the NFVI services impact the VNFs that are being hosted and also reflect the inherent nature of the services being offered by the NFVI, that is Virtual Machines (including virtual storage) and Virtual Networks. As such they need to be understandable by and visible to both the NFVI and the VNFs (or the VNF Managers) In addition, these quantitative measures should make maximum practical usage of existing industry standard quality measures in order to leverage both telecommunication operators' and suppliers' expertise in this area.

As Key Quality Indicators (KQIs) are developed, additional considerations need to be taken into account:

1) Specifications of new metric/indicator definitions and methods of measurement shall be clear and unambiguous.

2) Factors such as the ultimate use of each indicator, for design, or monitoring, SLA, or other purpose, should be identified.

3) The large scale of virtualized functions and networks encourages judicious selection of individual indicators.

4) Indicators selected should provide sufficient coverage of measurement scope or the subject element.

5) The indicators which are considered "key" for a specific scope/element can vary by according to implementation details and the operating administration. Fortunately, virtual architectures are flexible and KQIs can be removed/added as warranted.

6) Once the KQIs are determined, summary statistics and reporting formats can be selected, followed by on-going examination to ensure that inferences or assumptions continue to hold.

There is a natural tension between the desire for minimum indicators to accommodate large scale and coverage of all life-cycle aspects. However, there is a process which can help to address both considerations. Once the list of possible indicators for a subject or scope has been prepared, the candidate metrics can be organized according to the matrix in figure 20. Each cell of the matrix is associated with a life-cycle phase and a quality criterion (speed, accuracy, reliability). Some intersections are critical for certain functions or resources (and these would typically contain KQIs) while others may be inapplicable or contain secondary metrics.

	Speed	Accuracy	Reliability
Activation/Creation/Setup			
Operation			
De-activation/Deletion/Take-down			

Figure 20: Quality Indicator Matrix

By listing each quality indicator in its corresponding cell, it is possible to identify overlaps and gaps.

The matrix may also facilitate the process to determine which quality indicators are the key ones and should be measured, collected, and reported. Usually designers make the KQI designation based on their initial understanding of the system and its limitations, but there is no substitute for operational experience. For example, VM activation time would be considered a KQI in most circumstances, but VM activation reliability could be an equally important symptom of infrastructure issues under production load. The matrix of Quality Indicators can clearly indicate which aspects of performance are monitored, and which aspects are secondary (if measured at all).

Each Use Case presents its own unique demands on the NFV system, therefore it is important to revisit the matrix composition for each case to ensure that KQIs are present or omitted as needed.

6.7 Security Aspects

Security of the NFVI is best addressed in a divide-and-conquer fashion according to the defined domains. In general, security of infrastructure network domain, hypervisor domain, compute domain and network application domain can leverage the applicable security guidelines outlined in the existing standards development organizations and industry forums. Some of these organizations are IETF, IEEE, 3GPP, ISO/IEC JTC1, 3GPP, CSA, ATIS and ITU-T.

For most part security has been considered in the different components already, but independently from each other. Typically, network functions are standardized with respective security considerations. For example, consider IMS with the different functional entities, such as P-CSCF, S-CSCF, and HSS. The protocols for these entities have been standardized in the 3GPP and the IETF, and their built-in security mechanisms should be re-used.

For the hypervisor and operating systems of network functions, the type of security investigations differ from the communication security aspects typically considered in standards organizations like ETSI, 3GPP, and IETF. In general, attention has to be paid to secure coding, security testing, access control (to prevent unauthorized access and escalation of privileges), hardening, and timely patch management.

Network function virtualization increases the attack surface. In a traditional telecom environment, it is sufficient to secure the hosts (with their operating systems), the applications running on these hosts, and the communication between these applications. With network function virtualization, it is also necessary to protect the hypervisor and its communication with the management infrastructure, and employ strict identity and access management. An unsecure hypervisor will impede adequate isolation of VMs running on the same host, inadequate identity and access management will result in compromises of the whole NFV infrastructure. With that, unauthorized parties may maliciously or accidentally impact the lifecycle of virtual machines.

To ensure security of the infrastructure network domain, it is necessary to revisit the security assumptions made during the initial design of the network functions since they had often been made under different assumptions. For example, the threat models of certain protocols assume that adversaries are unable to gain access to the same LAN. Besides the basic security of the virtualized components or domains, a specific NFV use case may call for different kinds of security requirement. Thus, the security requirements for each of these use cases need to be studied separately. Some specific use cases of network application domain that are being discussed in ISG include virtualized EPC components (e.g. MME, S/P-GW, PCEF, PCRF), or virtualized IMS components (e.g. P-CSCF, I-CSCF, S-CSCF, HSS) running on shared VMs. Security of network application domain may be provided through mechanism such as security zoning, security gateway (VPNs), firewalls, ACLs, load balancing and dynamic DNS.

Protection of the NFV infrastructure necessitates threat management, which involves two aspects: visibility and control. Visibility refers to the ability to see and correlate information from the infrastructure to baseline proper behaviour and then to measure deviation from that norm. Control refers to the actions taken to mitigate an attack. Some controls are taken proactively while others are applied after an attack takes place. Threat intelligence can be obtained by applying proper filters to the system and network level data from the hypervisor, network-based detection, and other monitoring systems. Hypervisor-based introspection can help detect attacks on VMs and guest OS's, even when the guest OS's are tampered. Introspection is through monitoring of memory, program execution, access to data files, and network traffic. It can, in particular, thwart Kernel Level Rootkits (KLR). Regarding monitoring, If multiple monitoring systems (VM-based or not) are in use, they need to coordinated. Furthermore, monitoring may involve Deep Packet Inspection (DPI) to detect anomalies at a deeper level.

A result of network function virtualization is extensive use of APIs. If as much as control of the NFV infrastructure is done programmatically, there shall be strict API access control. In particular, it is essential that there be adequate security control in place when APIs are used to provide orchestration and interaction between virtualized network functions and the underlying infrastructure.

Data protection is an essential aspect of NFV security. It covers data confidentiality, data integrity, and access control. Various data protection techniques can be deployed based on use cases. In general, to protect data in motion (such as in VM migration or inter-VM communication or remote management), secure networking techniques such as TLS (Transport Layer Security), IPSec, or Secure Shell (SSH) can be applied. To protect data at rest, best practices for encryption and key management need to considered.

7 Domains of the NFV Infrastructure

This clause gives an overview of the architecture of each of the three domains of the NFVI. The detailed architecture of each domain are given in "NFV Infrastructure Architecture: Compute Domain" (ETSI GS NFV-INF 003 [i.12]), "NFV Infrastructure Architecture: Hypervisor Domain" (ETSI GS NFV-INF 004 [i.13]), and "NFV Infrastructure Architecture: Infrastructure Network Domain" (ETSI GS NFV-INF 005 [i.14]) respectively.

The NFVI comprises those elements needed to host VNFs. The NFVI therefore includes all hardware components. The NFVI also including some software components which are either common to many VNFs and/or provide functionality required to support the deployment, interconnection, or management VNFs.

The NFVI is deployed as one or more NFVI-Nodes which collectively implement the required functionality to support the execution environment for VNFs. Figure 21 provides a comparison of the physical and functional views of NFVI. Each NFVI-Node figure 21(b) may implement some amount of capacity of the functions identified in figure 21(a).

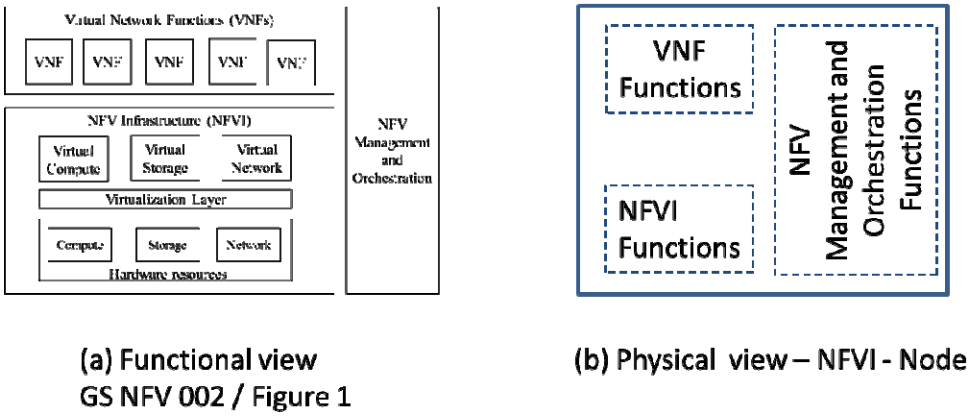

Figure 21: Comparing functional and physical views of NFVI

A location where an NFVI-Node is deployed is an NFVI-PoP. An NFVI-PoP may contain one or more NFVI-Nodes as well as other network elements. The Architectural Framework provides an example figure 2/ETSI GS NFV 002 [2] of an end-end network service implemented by three network functions (Nes) interconnected by infrastructure networks. Figure 3/ETSI GS NFV 003 [3] provides a corresponding end-end service implemented using VNFs. There is no specific mapping of the end points and physical or virtual network functions to specific locations, but in general the NFs could be in separate locations as could the VNFs. VNFs in separate NFVI-PoPs would be in separate NFVI nodes. Figure 22 shows a functional view (b) corresponding to the physical view (a) provided in Figure 2/ETSI GS NFV 002 [2]. The physical view of the corresponding VNF service graph (where all the NFs are completely virtualized) is provided in figure 3/ETSI GS NFV 002 [2] and reproduced here as figure 22(c). In general, end-end services based on VNF service graphs can include both NFs and VNFs, and the VNFs would be implemented in NFVI-Nodes. Figure 22(d) shows this functional view of the general case where NFVI-Nodes, existing NFs (Nes) and End Points are interconnected by Infrastructure Networks.

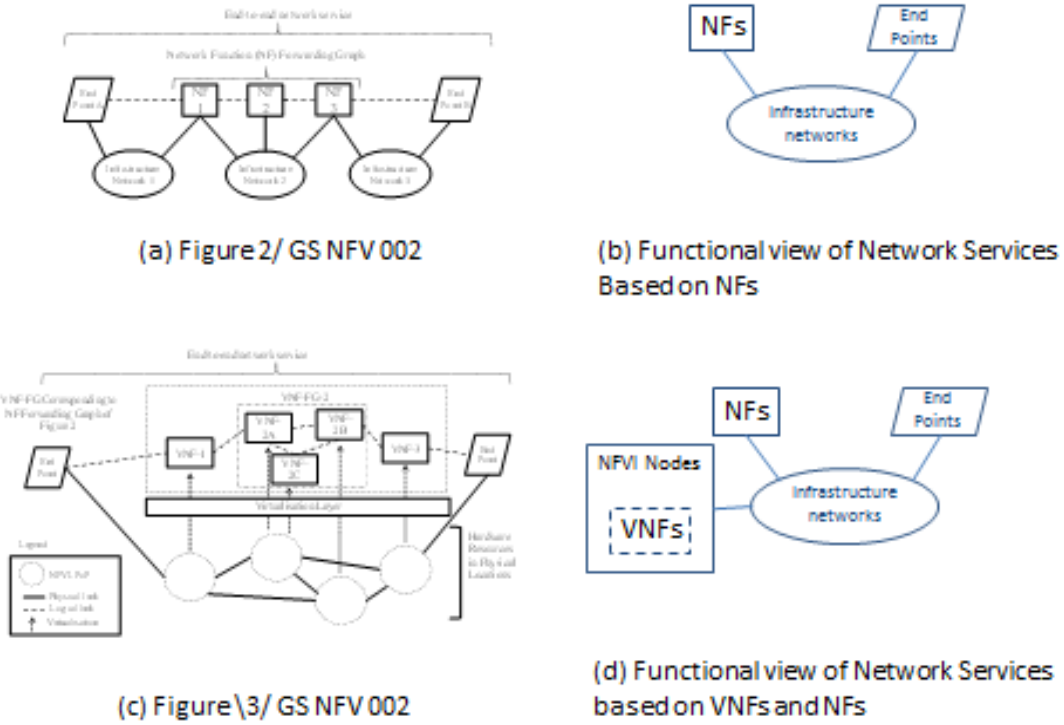

Figure 22: Comparing physical and functional views of the example from ETSI GS NFV 002 [2]

The number of NFVI-Nodes that a service provider will require depends on the capacity of those nodes in comparison to the workload as well as the field of application. ETSI GS NFV 001 [1] identifies fields of application with quite different requirements for the number of NFVI-Nodes. Use cases #1-4 may be deployable in a few relatively centralized NFVI-Nodes (though they could also be more widely distributed). Use cases #5-9 consider the virtualization of more widely distributed network functions (e.g. mobile base stations, CDNs, fixed access network functions). These use cases may imply a much larger number and more widely distributed set of NFVI-Nodes.

In order to control the complexity of the infrastructure, it is divided into three domains as follows:

- the compute domain;
- the hypervisor domain;
- the network infrastructure domain.

These domains allow for largely autonomous supply and evolution of each domain. Given this purpose, the boundaries are not precisely defined and there is some functional overlap between the domains.

NFV is applicable to any data plane packet processing (see note) and control plane function in fixed and mobile network infrastructures. NFV can also be used to provide an efficient production environment which can commonly be used by different applications, users and tenants, supporting the coexistence of several versions and variants of a network service (including test versions and beta versions). NFV involves the implementation of network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new custom hardware equipment. NFV aims to leveraging standard IT virtualisation technology to consolidate many network equipment types onto industry standard high-volume Commercial Off-The-Shelf (COTS) servers, switches and storage, which could be located in Datacentres, Network Nodes and in the end user premises.

NOTE: Data plane packet processing can include packet forwarding, packet header manipulation, as well as possible manipulation of the packet payload, for example for transcoding.

Network operators need to be able to "mix & match" COTS hardware from different vendors, without incurring significant integration costs and avoiding lock-in and eliminating the need for application-specific hardware. The skills base across the industry for operating standard high volume IT servers is much larger and less fragmented than for today's telecom-specific network equipment.

Figure 23 gives a high level overview of the three domains within the NFVI and shows how the domains realize the primary interfaces of the NFV overall architectural framework (see figure 2).

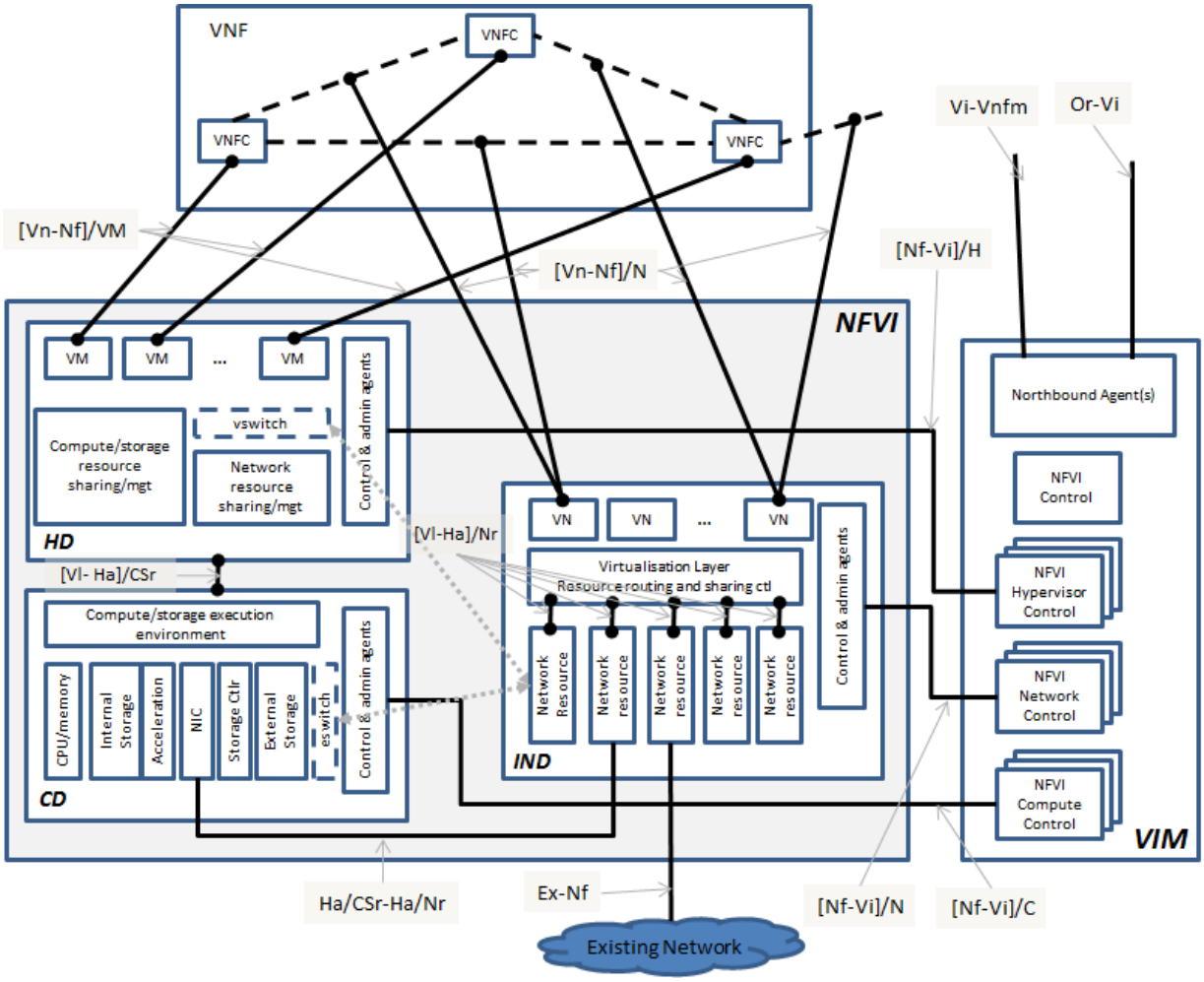

Figure 23: High level overview of the NFVI domains and interfaces

The characteristics of each reference point identified in figure 23 together with their relationship with the reference points of figure 2 and interfaces of figure 9 are set out in table 2.

Table 2: Characteristics of NFV Infrastructure Reference Points

INF Context	NFV Framework Reference Point	INF Reference Point	Reference Point Type	Correspondence with figure 9 Interfaces	Description and Comment
External	Vn-Nf	[Vn-Nf]/VM	Execution Environment	7	This reference point is the virtual machine (VM) container interface which is the execution environment of a single VNFC instance.
		[Vn-Nf]/N	Execution Environment	6	This reference point is the virtual network (VN) container interface (e.g. an E-Line or E-LAN) which carrying communication between VNFC instances. Note that a single VN can support communication between more than a single pairing of VNFC instances (e.g. an E-LAN VN).
	Nf-Vi	[Nf-Vi]/N	Management, and Orchestration Interface	9	This is the reference point between the management and orchestration agents in the infrastructure network domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the infrastructure network domain.
		[Nf-Vi]/H	Management, and Orchestration Interface	10	This is the reference point between the management and orchestration agents in hypervisor domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the hypervisor domain.
		[Nf-Vi]/C	Management, and Orchestration Interface	11	This is the reference point between the management and orchestration agents in compute domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the compute domain.
	Vi-Vnfm		Management, Interface	5	This is the reference point that allows the VNF Manager to request and/or for the VIM to report the characteristics, availability, and status of infrastructure resources.
	Or-Vi		Orchestration Interface	5	This is the reference point that allows the Orchestrator to request resources and VNF instantiations and for the VIM to report the characteristics, availability, and status of infrastructure resources.
		Ex-Nf	Traffic Interface	13	This is the reference point between the infrastructure network domain and any existing and/or non-virtualised network. This reference point also carries an implicit reference point between VNFs and any existing and/or non-virtualised network (Interface 1 of figure 9).
Internal	Vl-Ha	[Vl-Ha]/CSr	Execution Environment	12	The framework architecture (see figure 2) shows a general reference point between the infrastructure 'hardware' and the virtualisation layer. This reference point is the aspect of this framework reference point presented to hypervisors by the servers and storage of the compute domain. It is the execution environment of the server/storage.
		[Vl-Ha]/Nr	Execution Environment		The framework architecture (see figure 2) shows a general reference point between the infrastructure 'hardware' and the virtualisation layer. While the infrastructure network has 'hardware', it is often the case that networks are already layered (and therefore virtualised) and that the exact choice of network layering may vary without a direct impact on NFV. The infrastructure architecture treats this aspect of the Vi-Ha reference point as internal to the infrastructure network domain.
		Ha/CSr-Ha/Nr	Traffic Interface	14	This is the reference point between the infrastructure network domain and the servers/storage of the compute domain.

The following give further expansion and clarification to the Infrastructure domains and reference points defined in figure 23 and table 2.

- ***Layering and composition of container interfaces (execution environments).*** The fundamental architecture of virtualisation described in clause 6 can be composed and layered. The virtualisation of the compute and storage resources into virtual machines can be combined with the virtualisation on network resources in network services (e.g. E-Line and E-LAN) can be combined and provide a single composite container interface for a complete VNF. This composite container interface is an inherently parallel and communicating execution environment.

- ***Treatment of a virtual switch (vSwitch) and server embedded switch (eSwitch).*** These functions are clearly implemented in their respective domains and as such are proper part of those domains. However, functionally, they form an integral part of the infrastructure network domain. The functional specification of these switches is considered to be part of the infrastructure network domain. However, from the implementation point of view, they are considered to be part of the hypervisor domain and the compute domain respectively.

- ***Partitioning of Management and Orchestration Functionality.*** The framework architecture divides management and orchestration functionality between the Orchestrator, the VNF Manager, the VIM, and the management and orchestration agents with the infrastructure domains. The architecture identifies the existence of a clear hierarchy of orchestration and management, however, the exact split of functionality within this hierarchy may vary. Guidance on the scope of the management and orchestration agents in each domain is considered in the architecture of each domain.

7.1 Compute Domain

The role of the compute domain is to provide the COTS computational and storage resources, when used in conjunction with a hypervisor of the hypervisor domain, needed to host individual components of VNFs. Broadly speaking, the compute domain provides the interface to the network infrastructure domain, but does not support network connectivity itself.

Reduced hardware equipment costs and reduced power consumption through consolidating equipment are expected through exploiting the economies of scale of the IT industry. Reduced energy consumption by exploiting power management features in standard servers and storage, as well as workload consolidation and location optimization. For example, relying on virtualisation techniques it would be possible to concentrate the workload on a smaller number of servers during off-peak hours (e.g. overnight) so that all the other servers can be switched off or put into an energy saving mode.

Network Functions Virtualisation will leverage modern technologies such as those developed for cloud computing. At the core of these cloud technologies are virtualisation mechanisms: virtualisation of hardware by means of hypervisors, as well as the usage of virtual Ethernet switches (e.g. vSwitch) for connecting traffic between virtual machines and physical interfaces. For communication-oriented functions, high-performance packet processing is available through high-speed multi-core CPUs with high I/O bandwidth, the use of smart Ethernet NICs for load sharing and TCP Offloading, and routing packets directly to Virtual Machine memory, and poll-mode Ethernet drivers (rather than interrupt driven, for example Linux NAPI and Intel's DPDK).

The use of industry standard high volume servers is a key element in the economic case for NFV. Network Appliances which depend on the development of bespoke Application Specific Integrated Circuits (ASICs) will become increasingly uncompetitive against general purpose processors as the cost of developing ASICs increases exponentially with decreasing feature size. NFV leverages the economies of scale of the IT industry. An industry standard high volume server is a server built using standardized IT components (for example x86 or ARM architectures) and sold in the millions. A common feature of industry standard high volume servers is that there is competitive supply of the subcomponents which are interchangeable inside the server.

The principal elements of the compute domain are illustrated in figure 24. While this figure shows physical components, this does not mean that NFV defines the way the compute infrastructure is implemented. The diagram is illustrative in this regard.

- The CPU. This is the generic processor which executes the code of the VNFC.

- The Network Interface Controller (NIC). This provides the physical interconnection with the infrastructure network domain (Interface 14 of figure 9).

- Storage. This is large scale and non-volatile storage. In practical implementation, these include spinning disks and solid state disks (SSDs). The functional definition of storage and its coupling to the 'unit of compute' is discussed further below.
- Server. This is the logical 'unit of compute' and is this basic integrated computational hardware device; in practical implementation, a 'server'. The functional constraints rather than practical implementation constraints which properly and robustly determine the boundaries of discrete 'units of compute' are discussed further below.
- Chassis. Practical housing of compute hardware. This should be essentially independent and transparent to the users of the compute domain.
- Remote management. This is management specific to the compute domain and is largely transparent to the users of the compute domain.

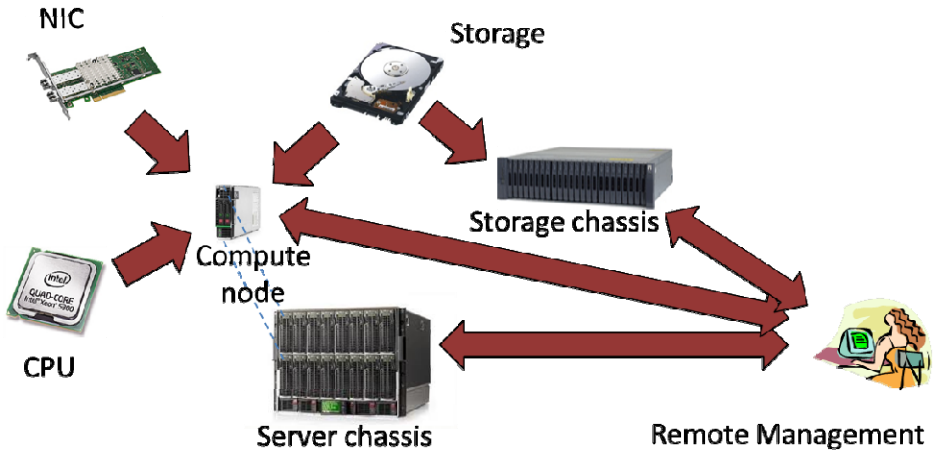

Figure 24: Elements of the compute domain (for illustrative purposes)

7.1.1 Functional Description of the Compute Domain

This illustrative description of the compute domain has many features which are specific to certain implementations are not necessary to the definition. Specifically, it is possible to have other implementations which do not fit with this implementation based definition yet fully meet all the requirements of the compute domain.

As an example, a key part of the compute domain is the processor instruction set. One might reasonably expect this to be implemented by a piece of hardware called a CPU. However, there are many examples where a processor instruction set is emulated and run by a processor of a different instruction set. Functionally these two implementations may be exactly equivalent. Where these two implementations may differ in properly functional terms is in the cycle time required to execute each instruction as time is still proper to a functional description. A further insight into the nature the functional description can be gained by considering that many hardware implemented CPU are still themselves translating the instruction set to microcode.

In summary, what matter for the NFV description is the logic of the execution and how fast the execution occurs.

7.1.2 Compute Node

At a practical level, a server is a local and the small physical implementation is important to the nature of a server. As an immediate illustration, most applications, including base elements implemented within a single VNFCs and hence within a single VM, cannot normally be implemented on a distributed cluster of servers. However, this description of this primary constraint on the compute domain architecture is described in implementation terms which is not appropriate for objectives of the ISG.

To meet with the objectives of the ISG, it is necessary to turn this well understood practical definition into one which is functionally defined and is not dependent on implementation details.

In line with the functional description of the compute domain, this needs to reference the speed of execution. The unit of compute is therefore defined as follows.

- **Compute Node.** A compute node is a functional entity which is capable of executing a generic computational instruction set (each instruction be being fully atomic and deterministic) in such a way that the execution cycle time is of the order of units to tens of nanoseconds irrespective of what specific state is required for cycle execution.

In practical terms, this defines a compute node in terms of memory access time. A distributed system cannot meet this requirement as the time taken to access state stored in remote memory cannot meet this requirement.

7.1.3 Functional Description of Storage

In functional terms, storage holds state.

Therefore the primary characteristics of storage are:

- The latency in accessing a specific state held in storage in order to execute a instruction cycle.
- The size of the storage.
- The volatility or persistence of the storage.

Storage in the compute is also normally strongly associated with a specific technology. Examples include:

- CPU registers.
- CPU cache.
- Volatile RAM.
- Non volatile block storage, e.g. spinning disks and solid state disks.

Each of these are characterized by different levels of latency in state access for instruction cycle execution which is normally is trade-off with the size of the storage. For some purposes, it is necessary to have visibility of the different types of storage. However, for many applications, the different forms of storage can be abstracted, especially when one form of storage is used to cache another form of storage. For example:

- CPU cache holding a transparent cache of part of the volatile memory.
- Volatile memory only hold part of a machine's address space using a page allocation system.
- Volatile memory working in conjunction with swap space on a disk to increase the number of allocated pages to a machine.

Even though the latency of a specific storage may not come within the definition above for a unit of compute, when used in these contexts, the overall effect is normally to come within the definition.

Persistent storage may be attached locally to the compute node but is also frequently attached via Storage Area Networks (SANs) (see note). Large redundant storage arrays can be attached to multiple compute nodes via SANs which provide high-speed block level access to the storage such that allocated capacity in the storage array can appear to the compute node as if it is local storage. It is often the case that the latency of the SAN does not greatly increase the latency already inherent in the disk access.

NOTE: LANs can also be used for access to storage arrays. There is a general and gradual convergence between SAN protocols and their associated technology and the Ethernet protocols and technology of LANs.

7.1.4 Scope of a Compute Node

While the compute node is defined in term of execution of a generic instruction set, any compute node will have more functionality than just a CPU with storage. At minimum, the scope will include some form of NIC.

In addition, the scope can include other 'acceleration' hardware. Examples include:

- hardware for encryption and decryption;
- packet switching, for example within a NIC;
- 'accelerated' packet forwarding.

These can normally be plugged into the standard server hardware using, for example, a PCIe bus or on a System-on-Chip.

However, the use of such acceleration by VNFCs will reduce the portability of the VNFCs and there will be a trade-off between the cost of prior provision of the accelerator hardware, the flexibility of deployment, and the value of the acceleration.

7.1.5 Standardizing Organisations Impacting the Compute Domain

Existing industry bodies working on Cloud Computing Infrastructure of potential relevance to NFV infrastructure include:

- The Organization for the Advancement of Structures (OASIS) has a number of on-going technical initiatives related to Cloud including architectures, protocols, management of Cloud platforms, etc. In some cases the NFV virtualization infrastructure could be considered as a PaaS as described in clause 6.2.2.2. Further studies of the compute nodes should consider the applicability of OASIS specifications.
- The TM Forum (TMF) has on-going work on cloud as part of their digital services initiative. This includes requirements for enterprise grade IaaS compute infrastructures [i.5]. As discussed in clause 6.2.2.3, the Virtualization infrastructure could be considered an IaaS platform on which the NFV software applications execute. Further studies on the compute nodes of the virtualization infrastructure should consider the applicability of industry specifications for IaaS requirements.
- The Distributed Management Task Force (DMTF) has on-going technical work on Cloud Management Standards Their technical specifications include XML data formats such for the open, secure, portable, efficient and extensible packaging and distribution of software to be run in virtual machines [i.6]. The virtualized network functions are software that will be packaged and distributed. These VNFs should be able to be executed on both virtual machines and in some cases on bare metal compute nodes. Further studies of Compute nodes should give consideration as to the file formats that the virtualisation Infrastructure should support and the applicability of existing cloud management specifications for the management of cloud infrastructure such as NFV compute nodes.
- The Storage Networking Industry Association (SNIA) has on-going technical work considering the virtualization of storage in the cloud as part of their Cloud Storage Initiative. SNIA has also developed a number of technology standards to enhance the interoperability of storage systems, including some directed towards cloud type infrastructures. The compute nodes of the virtualization infrastructure include various aspects of storage for both general computational needs as well as supporting certain virtualized network functions with more intensive storage needs e.g. CDNs. Further studies on Compute Nodes should consider the applicability of the currently available storage specifications from SNIA to NFV.
- The Open Data Center Alliance (ODCA) has ongoing technical work considering usage models for cloud services in an open, multisource context. Their cloud service models range through IaaS, PaaS and SaaS and include public, private, hybrid and community cloud delivery models. Their work on Compute Infrastructure as a Service [i.7], Service Level Agreements [i.9], and Cloud Interoperability [i.8] may be relevant during further consideration of the NFV Compute Domain.
- Open Compute Project (OCP) has a mission is to design and enable the delivery of the most efficient server, storage and data centre hardware designs for scalable computing defined with open interfaces between server components. The Open Compute Project Foundation provides a structure in which individuals and organizations can share their intellectual property with Open Compute Projects.

7.2 Hypervisor Domain

The hypervisor domain mediates the resources of the computer domain to the virtual machines of the software appliances. Hypervisors as developed for public and enterprise cloud requirements place great value on the abstract they provide from the actual hardware such that they can achieve very high levels of portability of virtual machines.

In essence, the hypervisor can emulate every piece of the hardware platform even in some cases, completely emulating a CPU instruction set such that the VM believes it is running on a completely different CPU architecture from the actual CPU on which it is running. Such emulation, however, has a significant performance cost. The number of actual CPU cycles needed to emulate virtual CPU cycle can be large.

Even when not emulating a complete CPU architecture, there can still be aspects of emulation which cause a significant performance hit. For example, when more than one virtual machine is running on a single core, even with the same CPU architecture for the VMs, the hypervisors will normally run a multi-tasking executive in order to share the native CPUs cycles between the VMs. Each 'context switch' of the actual CPU's single thread between VMs can have a significant performance hit.

In addition, as there are many virtual machines all running on the same host machine, they will all reasonably want to interconnect with each other. The hypervisor provides emulated virtual NICs for the VMs, however, there is also the need for a virtual Ethernet switch to provide connectivity between the VMs and between each VM and the actual NICs. This is the hypervisor vSwitch. The vSwitch may also be a significant performance bottleneck.

The general architecture of a cloud hypervisor is shown in figure 25.

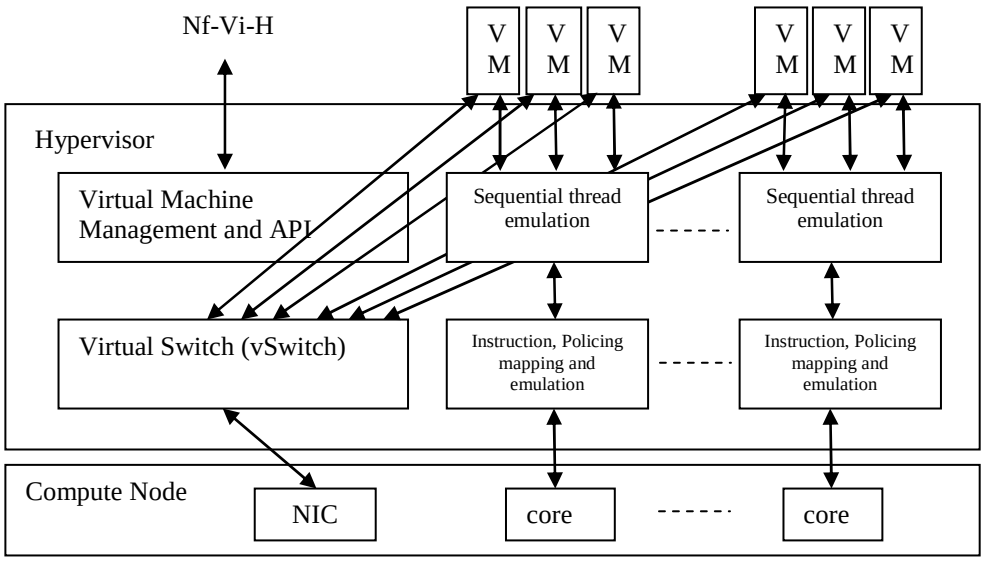

Figure 25: General public and enterprise cloud hypervisor architecture

For many NFV applications, performance is important. This means that many NFV virtual machines are required to run as fast as is practical. There are a number of features available in current and immediately forthcoming server hardware which greatly improve the performance of VMs. These include:

- multicore processors supporting multiple independent parallel threads of execution;
- system-on-chip processors that integrate multiple cores, DRAM interfaces, network interfaces, storage interfaces and hardware acceleration for security, multicore processing, networking, storage and application acceleration;
- specific CPU enhancements/instructions to control memory allocation and direct access on I/O devices to VM memory allocations;
- PCIe bus enhancements, notably SR-IOV.

These allow high performance VMs to run effectively as if they are running natively on the hardware, however, they are still under the full control of the hypervisor. Specific features of the hypervisor support for high performance NFV VMs include:

- exclusive allocation of whole CPU cores to VMs;
- direct memory mapped polled drivers for VMs to directly access the physical NICs using user mode instructions requiring no 'context switching';
- direct memory mapped polled drivers for inter VM communications again using user mode instructions requiring no 'context switching';
- vSwitch implementation as a high performance VM again using direct memory mapping and user mode instructions requiring no 'context switching'.

The resulting hypervisor architecture is one the primary foundations of the NFV infrastructure. The NFV hypervisor architecture is shown in figure 26.

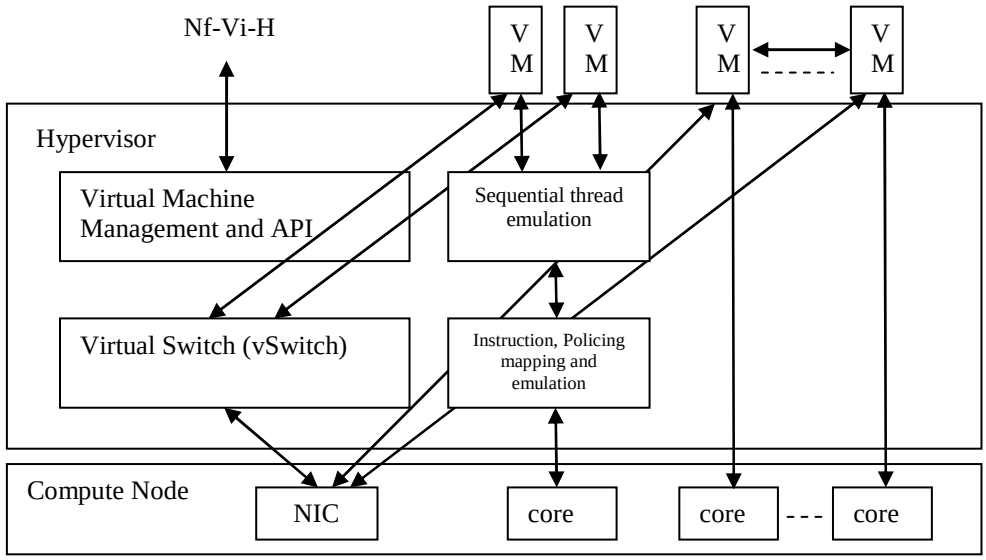

Figure 26: NFV hypervisor architecture

Figure 26 shows the NFV hypervisor architecture and is largely defining of the overall NFV architecture. It is one of a relatively small number of critical components which enable the objectives of NFV to be met. This hypervisor architecture provides all the performance of 'bare metal' which also providing the full orchestration and management provided by the hypervisor.

A possible interim and non-preferred alternative which can still provide some of the benefits of NFV is to accept a significantly reduced modularly of hardware resource as being the server itself and then dedicate a whole server to a software appliance module. This would also rely on the server to provide the means of remote install and management. While most servers do provide such an interface, they are often proprietary and even when standards are used, these interfaces are not the primary focus of the orchestration and management systems, especially as developed as part of cloud technologies. However, such an arrangement could still offer significant advantages over custom hardware and also provide a reasonable migration path to the full NFV architecture with NFV hypervisors.

7.2.1 Standardizing Organisations Impacting the Hypervisor Domain

Existing industry bodies working on Cloud Computing Infrastructure of potential relevance to NFV infrastructure include:

- The Open Data Center Alliance (ODCA) has ongoing technical work considering usage models for cloud services in an open, multisource context. Their cloud service models range through IaaS, PaaS and SaaS and include public, private, hybrid and community cloud delivery models. Their work on VM interoperability and software entitlement management should be considered during further studies on Infrastructure Software such as hypervisors.

7.3 Infrastructure Network Domain

The infrastructure network domain performs a number of roles. It provides:

- the communication channel between the VNFCs of a distributed VNF;
- the communications channel between different VNFs;
- the communication channel between VNFs and their orchestration and management;
- the communication channel between components of the NFVI and their orchestration and management;
- the means of remote deployment of VNFCs;
- the means of interconnection with the existing carrier network.

The first of these are the basic connectivity services which are hosted by the infrastructure container interface on behalf of the VNFs. The role of the infrastructure is to provide a large number of discrete connectivity service instances. In general, it is a fundamental requirement that these services are separate from each other and not interconnectable with each other. This is the antithesis of the universal networking of the Internet or the PSTN.

Figure 27 illustrates one particular scenario which demonstrates the essential requirement of the ***disconnectedness*** between the connectivity services. In this scenario, the VNF is composed of multiple client VNFCs which access application VNFCs through multiple gateway VNFCs. In this case, any connectivity between the access connectivity service and the core connectivity service would completely bypass and defeat the role of the gateway VNFCs. Moreover, any connectivity between different access connectivity services would also undermine any client specific aspects of the gateway VNFCs.

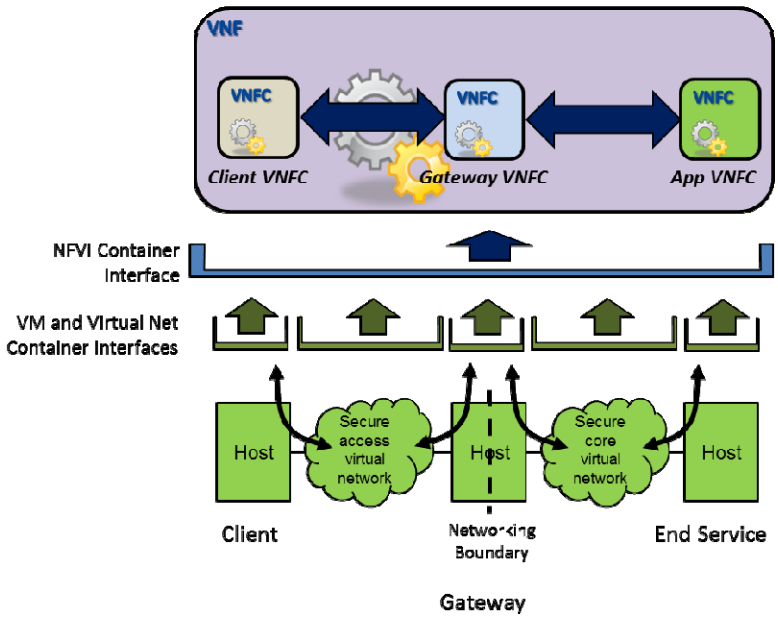

Figure 27: Security requirements for the infrastructure network services

As a direct result of the infrastructure network roles identified above, the infrastructure network shall pre-exist any VNFs but shall provide sufficient network connectivity to carry out its roles. In order to achieve this, the infrastructure network shall embed the essential elements needed to provide connectivity already self-contained with the domain. These include:

- an infrastructure addressing scheme (there may well be more than one scheme) with address allocation and management;
- a routing process which can relate infrastructure addresses to routes through the infrastructure network topology;
- a bandwidth allocation process;

- a set of OAM processes to verify reliability, availability, and integrity of connectivity services.

While, the resource management process could be as simple as a packet egress FIFO queue with tail drop, this is unlikely to satisfy the requirement of most NFV use cases. For example, the requirement for the bandwidth aspects of the communications services for some VNFs may involve some or a mixture of the following:

- permanently allocated, fully guaranteed bandwidth;
- respecting of prioritization markings of packets should packets be dropped;
- guaranteed minimal latency variation;

All of which are affected by the way bandwidth resource is allocated in the infrastructure network.

While the VNFs may run integrity verification processes across the connectivity services as part of their own protocols, these protocols shall be transparent to the infrastructure network. Therefore the infrastructure network, in order to manage and maintain the infrastructure network services and operate to any service level agreements, shall run its own set of OAM protocols. These need to be able to, for example:

- detect any general loss of connectivity;
- detect any loss, miss-insertion, or miss-delivery of traffic;
- detect any excessive variation in latency.

The requirement that the infrastructure network and the VNF protocols shall be independent and transparent to each other implies that there are three issues to be resolved between the infrastructure network and the VNFs:

- ***Common Header*** - this will be a MAC header associated by the virtual NIC presented to the VNFC by the hypervisor. The MAC addresses of the virtual NICs are supplied by the infrastructure and are visible to and understood by both the infrastructure and the VNFCs.
- ***Transparent encapsulation*** - the infrastructure needs to carry the packets from the VNFC as a transparent payload and shall add a completely new header for transport by the infrastructure network.

NOTE: The simplest transparent encapsulation is a VLAN, however, neither IEEE 802.1Q [i.18] or even IEEE 802.1ad [i.19] are likely to scale to the needs of the NFV.

- ***Address binding*** - as the infrastructure network has its own addressing and routing scheme, the MAC addresses presented to the VNFC shall be bound to infrastructure network addresses.

Finally, NFV architecture acknowledges that there will be more than one technology solution to infrastructure network. This recognizes that there are different solutions already in use and which will persist. Therefore excluding network technologies which are already widely deployed would severely limit the deployment of NFV solutions. However, this plurality of infrastructure networking solutions creates an interworking requirement.

This interworking requirement is illustrated in figure 28. The primary requirement being that the overall infrastructure network container interface hosts connectivity services seamlessly.

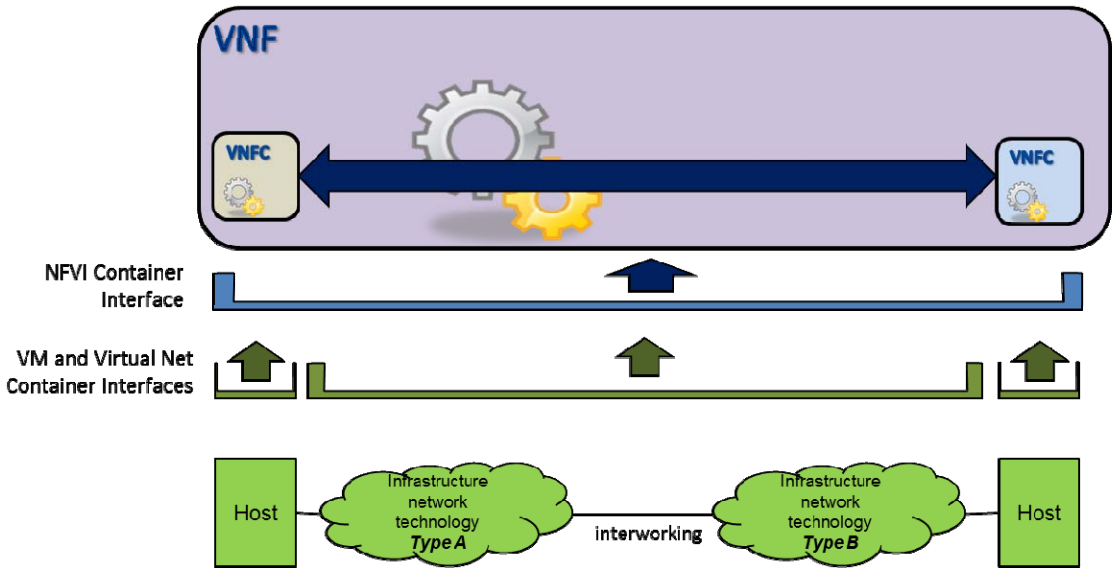

Figure 28: Interworking requirement for the infrastructure network

7.3.1 Standardizing Organisations Impacting the Infrastructure Network Domain

Existing industry bodies working on Cloud Computing Infrastructure of potential relevance to NFV infrastructure include:

- The Open Data Center Alliance (ODCA) has ongoing technical work considering usage models for cloud services in an open, multisource context. Their cloud service models range through IaaS, PaaS and SaaS and include public, private, hybrid and community cloud delivery models. Their work on IO Control for passing QoS guarantees [i.9] from the system to the network should be considered in further studies on the Infrastructure network.
- The Internet Engineering Task Force (IETF) has ongoing technical work developing network protocols that should be considered in further studies on the Infrastructure Network. Relevant working groups include FORCES, NETMOD, NETCONF, ALTO, NVO3, and I2RS.
- IEEE 802 is responsible for the standardization of Ethernet and related protocols.
- Metro-Ethernet Forum (MEF) has already specified Ethernet service including E-Line and E-LAN which are directly relevant to Infrastructure Connectivity Services.
- Recommendation ITU-T SG15 [i.22] specifies standards relevant to some access and transport networks as well as the abstract network models on which their management interfaces are based.
- Broadband Forum has many specifications especially of access networks.
- TMForum have relevant specifications of management interfaces with the infrastructure network domain.

8 Challenges in Performance and Portability of VNFs

One of the main objectives that NFV addresses is making possible that virtual network functions can provide high and predictable performance while being portable between servers and hypervisors. This means maximizing the performance of SW-based network functions (VNFs) while achieving their effective independence from the hardware on which they will run.

This objective requires facing four big challenges to maximize performance and guarantee portability between servers.

8.1 Challenge 1: Processing performance depends on a number of factors

Unsurprisingly, the underlying HW server characteristics have a deep impact on the performance. Parameters such as processor architecture, clock rate, size of the internal processor cache, memory channels and speed, memory latency, bandwidth of inter-processor buses and peripheral buses, etc. have a strong impact on the performance of the specific application or VNF running on that HW.

However, SW design shall be taken into account, as it is of utmost relevance to take advantage of all capabilities offered by multi-core processor architectures. Aspects such as pipelining, multi-threading and deadlock avoidance should be part of any VNF, and this is even more essential when dealing with data plane VNFs.

SW shall be oriented by design to achieve the highest possible performance in a multi-core architecture. Then, virtualization can come into play to increase performance by allowing the scale through different hosts. Some of the general recommendations to apply to network SW design and, particularly, to VNFs dealing with data plane workloads are the following:

- SW requiring high performance should be partitioned in independent modules running in different threads.
- Modules should have independent memory structures in order to avoid OS deadlocks.
- Modules should communicate each other in pipeline structures, minimizing the use of the OS as an intermediate step.
- OS interruptions should be reduced, if possible, to zero, due to its high consumption of CPU cycles.
- The use of OS network stacks should be avoided. The intelligence of the network stack is to be provided by the modules themselves, which should read and write directly from/to memory without the help of the OS.
- Network cards and other peripherals should communicate directly with the modules through memory, making use of technologies such as DMA and its virtualized forms.

Note that the proliferation of virtual machines as the first option to increase performance of a VNF is discouraged as a general rule. In some situations, VM proliferation might be presented as the solution to scale SW-based nodes through the sort of parallelization that these technologies provide. However, the intensive (ab)use of these techniques might lead to unpredictable behaviours in scenarios where predictable linear scalability should be expected. Instead, it is recommended an adequate multi-threaded and pipelined SW design per VM, and limiting VM instantiation only to the purpose of dynamic scaling (via scale-out) of the function.

Finally, virtualization of network sub-functions and their automated resource allocation in a given pool of servers might require a different approach to the ones currently used in IT/cloud environments. Complex multi-threaded and pipelined SW designs, needed to achieve high performance, bring some difficulties when need to be deployed in a virtualized environment. In fact, the actual mapping of SW pieces onto HW elements is of utmost relevance. Different deployment combinations of a VNF over a given underlying HW may yield to radically different performance depending on how VNF modules are mapped to HW processor cores/threads. Sub-optimal mappings can lead to lower throughputs due to cache misses or saturation of the inter-processor bus.

8.2 Challenge 2: Interconnection of VNFs matter, and there are many options

When moving from the classical approach of interconnecting network functions to the NFV approach of VNF interconnection, different connectivity alternatives are available.

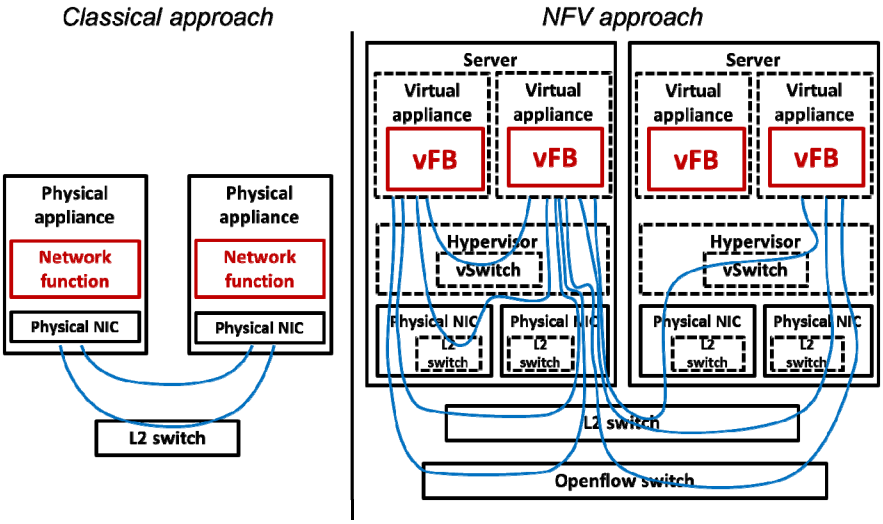

Figure 29: NFV introduces many more interconnection options

Some interconnection options rely on the switching capabilities of the hypervisor. Others bypass the Hypervisor or assume it does not exist, while relying on NIC capabilities (e.g. SR-IOV).

Each technique has its own advantages in terms of performance, flexibility or isolation. For instance, virtual interfaces offered by hypervisor provide lower performance in comparison to virtual interfaces built from "virtual functions" offered by SR-IOV-compliant NICs. However, hypervisor's virtual interfaces are still much simpler to configure than "virtual functions" and might support VM live migration in a natural way. Right option depends on the requirements of the VNF and the nature of the involved workloads (e.g. data plane vs. control plane).

8.3 Challenge 3: Virtualisation may bring portability at the expense of unpredictable performance

There are different ways to run network functions as SW-based network functions.

Running network functions directly on bare metal guarantees a predictable performance since, in principle, the HW mapping should be predictable. However, resource isolation is complex since different SW appliances will run as processes under the same OS and it may be difficult to restrict visibility between SW modules under the same OS. Moreover, the HW and OS view by the SW modules is, by nature, heterogeneous, since there is not a Hypervisor layer that can make it uniform regardless the HW and the OS. And, even more important, some appliances, that might have been designed to be run on a specific OS/kernel, could only be used if that is the underlying OS/kernel.

Virtualisation through a hypervisor improves portability significantly, making HW view uniform for virtual machines regardless the underlying HW. This also allows appliances to be run on the specific OS for which they were built without the need of using that specific OS as underlying OS. Furthermore, it provides a natural resource isolation (in terms of visibility), since now different VNFs can run as independent VMs and the hypervisor layer guarantees that there are no unexpected interactions between them. Nevertheless, in terms of performance, in order to guarantee resource isolation a very strict mapping of resources to SW modules needs to be done. In general, the use of a hypervisor layer may add some uncertainty to the performance as long as the VM does not control the exact mapping from the virtualized HW to the real HW. The pessimistic situation turns up in oversubscription scenarios where performance can decrease significantly if several VMs have an intensive usage of the oversubscribed resource (e.g. memory). This uncertainty due to the hypervisor can be technically avoided, but requires some tweaks in the hypervisor that may not be available out-of-the-box in current hypervisors.

Finally, OS-level virtualisation, based on containers or zones, provide a good control of resource fragmentation in oversubscription scenarios. However, they provide a loose mapping from the virtualized HW to the real HW, as well as loose resource isolation, so their use for data plane applications is still discouraged.

8.4 Challenge 4: The environment should be as simple to manage as possible

Virtual network functions should be deployed as simply as a drag and drop operation in an orchestration management system. In order to make this feasible, both VNFs and underlying HW should be described through some kind of abstraction (e.g. templates) that allows the automated matching by the orchestration management system.

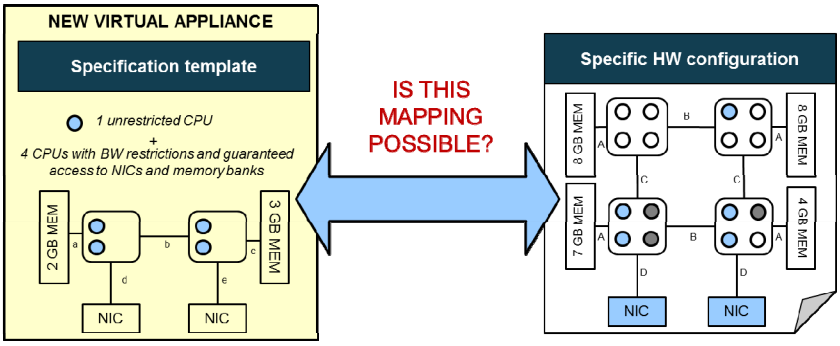

Figure 30: The challenge of a suitable abstract description of compute infrastructure

The definition and agreement on such an abstract description is a challenge that needs to be solved in the ISG to assure predictable performance and portability in a manageable and automated environment.

Annex A (informative): Contacts

Working Group Chairs:

- Mr Steven Wright, AT&T
- Mr Yun Chao Hu, Huawei

Working Group Managing Editor:

- Mr Andy Reid, BT

Rapporteur:

- Mr Andy Reid, BT

Andy Reid's work on the present document was part-funded by the European Union's Seventh Framework Programme FP7/2007-2013 under the Trilogy 2 project, grant agreement n°317756.

Annex B (informative): Bibliography

- USAGE: Standard Units of Measure For IaaS, Rev 1.1 2013, Open Data Center Alliance.

NOTE: Available at http://www.opendatacenteralliance.org/docs/Standard_Units_of_Measure_For_IaaS_Rev1.1.pdf.

History

Document history		
V1.1.1	January 2015	Publication