ETSI GS NFV-INF 003 V1.1.1 (2014-12)

GROUP SPECIFICATION

Network Functions Virtualisation (NFV); Infrastructure; Compute Domain

Reference

DGS/NFV-INF003

Keywords

NFV

Contents

Intellectual Property Rights

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.

Table 1: NFV infrastructure architecture documents

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 architectural description of the compute (& storage) domain of the infrastructure which supports virtualised network functions (VNFs). The compute domain includes the network & I/O interfaces required to interface to the infrastructure network and the storage network, if any.

It sets out the scope of the infrastructure domain acknowledging the potential for overlap between infrastructure domains, and between the infrastructure and the virtualised network functions. It also sets out the nature of interfaces needed between infrastructure domains and within the compute domain.

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).

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 referenced 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 GS NFV 001 (V1.1.1): "Network Functions Virtualisation (NFV); Use Cases".

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

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

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

[5] ETSI GS NFV-INF 001 (V1.1.1): "Network Function Virtualisation (NFV); Infrastructure Overview".

[6] DMTF DSP 0217: "SMASH Implementation Requirements".

[7] ETSI GS NFV-PER 001 (V1.1.1): "Network Function Virtualisation (NFV); NFV Performance & Portability Best Practices".

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 referenced 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] Master Usage Model: "Compute Infrastructure as a Service, Rev 1", (2012) Open Data Cneter Alliance.

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

[i.2] IEEE 802.3^{TM}: "Ethernet Working Group".

[i.3] ETSI GS NFV-REL 001: "Network Functions Virtualisation (NFV); Resiliency Requirements".

[i.4] IEEE 1588^{TM}: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems".

3 Definitions and abbreviations

3.1 Definitions

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

composite-NFVI: NFVI hardware resources are composed of field replaceable units that are COTS elements

field replaceable unit: unit of hardware resources designed for easy replacement during the operational life of a network element

gateway node: See ETSI GS NFV-INF 001 [5].

network node: See ETSI GS NFV-INF 001 [5].

NFVI components: NFVI hardware resources that are not field replaceable, but are distinguishable as COTS components at manufacturing time

NFVI-Plugin: NFVI hardware resources are deployable as a COTS field replaceable unit for another network element

NFVI-Pod: NFVI hardware resources are deployable as a single COTS entity with no field replaceable units

portability: See ETSI GS NFV-INF 001 [5].

storage node: See ETSI GS NFV-INF 001 [5].

3.2 Abbreviations

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

ACL	Access Control List
ACPI	Advanced Configuration and Power Interface
AES	Advanced Encryption Standard
API	Application Programming Interface
ARM	Acorn RISC Machine
ARP	Address Resolution Protocol

ASIC	Application Specific Integrated Circuit
AT&T	American Telephone & Telegraph
ATA	Advanced Technology Attachment
BBU	Base Band Unit
BIOS	Basic Input Output System
BRAS	Broadband Remote Access Server
BT	British Telecom
BW	Bandwidth
CBDMA	Common buffer DMA
CHD	Compute Host Descriptor
CIFS	Common Internet File System
CIM	Common Information Model
COTS	Commercial Off The Shelf
COW	Copy-On-Write
CPE	Customer Premise Equipment
CPU	Central Processing Unit
CRAN	Cloud Radio Access Network
CRC	Cyclic Redundancy Check
DAS	Direct Attached Storage
DCB	Data Center Bridging
DCMI	Data Center Management Interface
DMA	Direct Memory Access
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DSLAM	Digital Subscriber Loop Access Multiplexer
DSP	Digital Signal Processing
ECC	Error Correction Code
EMS	Element Management System
FC	Fibre-Channel
FCP	Fibre-Channel Protocol
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HDD	Hard Disk Drive
HSM	Hardware Security Module
HW	Hardware
HWA	Hardware Acceleration
IB	InfiniBand
IO	Input Output
IOMMU	Input Output Memory Management Unit
IOPS	Input Output Operations Per second
IP	Internet Protocol
IPC	Inter Process Communication
IPMI	Intelligent Platform Management Interface
ISA	Instruction Set Architecture
IT	Information Technology
KQI	Key Quality Indicator
KVM	Kernel Virtual Machine
LAN	Local Area Network
LLC	Limited Liability Corporation (?)
LTFS	Linear Tape File System
LUKS	Linux Unified Key Setup
LVM	Logical Volume Manager
MAC	Media Access Control
MIB	Management Information Base
MMU	Memory Management Unit
NAPT	Network Address and Port Translation
NAS	Network Attached Storage
NAT	Network Address Translation
NFS	Network File System
NFVI	Network Functions Virtualisation Infrastructure

NGFW	Next Generation Fire Wall
NIC	Network Interface Card
NPU	Network Processor Unit
OCP	Open Compute Project
ODP	Open Data Plane
OS	Operating System
OSPF	Open Shortest Path First
PCI	Peripheral Computer Interface
PCI-E	Peripheral Computer Interface - Enhanced
PGP	Pretty Good Privacy
RAID	Redundant Array of Independent Disks
RAN	Radio Access Network
RAS	Remote Access Server
RDMA	Remote Direct Memory Access
RIP	Routing Information Protocol
RoCE	RDMA over Converged Ethernet
SaaS	Software as a Service
SAN	Storage Area Network
SAS	SAN Attached Storage
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer Systems Interface
SDD	Solid State Disk
SDO	Standards Development Organization
SES	SCSI Enclosure Services
SLA	Service Level Agreement
SMASH	System Management Architecture for Server Hardware
SNMP	Simple Network Management Protocol
SR-IOV	Single Root Input Output Virtualisation
SSD	Solid State Disks
SSH	Secure Shell
SSL	Secure Socket Layer
SW	Software
TLB	Table Look-aside Buffer
TOR	Top of Rack
TPM	Trusted Platform Module
TX/RX	Transmit/Receive
UPS	Uninterruptable Power Supply
USB	Universal Serial Bus
VF	Virtual Function
VIA	Virtual Interface Architecture
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMD	VM descriptor
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFCI	Virtual Network Function Component Instance
VNFD	Virtual Network Function Descriptor
VNFM	Virtual Network Function Manager
VPN	Virtual Private Network
WAN	Wide Area Network

4 Domain Overview

Cloud computing in data centers have been able to abstract the hardware from the software through virtualisation reaping benefits from hosting multiple applications, accelerating time to market and offering over the top services. 64-bit multi-core processors with hardware support for virtual machines, form the core of the industry-standard server within the data center. These, coupled with offload and acceleration technologies, form the required server architecture. These are sometimes accelerated to meet workload performance. Data Center servers may use processors with 10 or more cores and have multiple sockets per server (e.g. a 4-way server with 10 cores has 40 physical cores).

NFV is an effort by the operators to leverage cloud computing benefits and enhance the abstraction of the software from the hardware for the network.

The present document will define the compute domain for NFV and document what is necessary for the compute domain to meet the NFV requirements.

4.1 Domain Scope

The Compute Domain is one of three domains constituting the NFV infrastructure, or NFVI. The other two domains are the Hypervisor, and the Infrastructure Network.

The compute domain includes storage & network, I/O interface - it comprises the generic servers and storage. It should be noted that the compute domain is closely associated with the orchestration and management domain. The latter runs on the NFVI as a set of modular, interconnected virtual machines.

4.2 High Level NFV Framework

The ISG NFV Architectural Framework document discusses three domains: VNFs; NFVI; and NFV Management & Orchestration. This is shown in the figure below depicting NFVI, which supports the execution of the VNFs including the diversity of physical resources and their virtualised counterpart. The Compute Domain is one of three domains constituting the NFVI. The other two domains are the Hypervisor, and the Infrastructure Network.

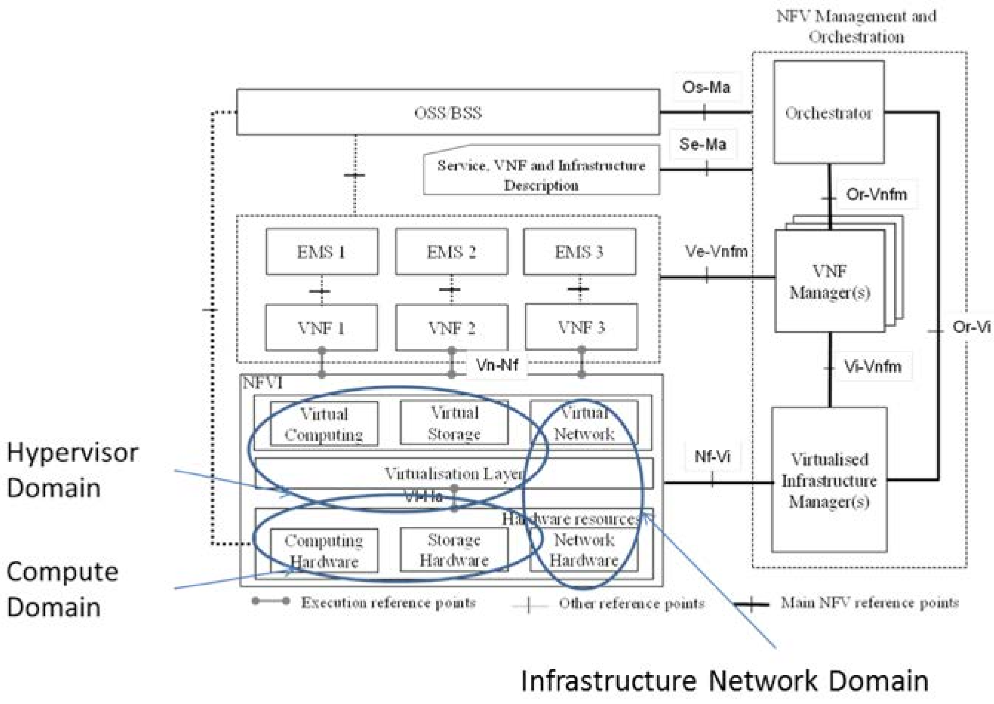

Figure 1: NFVI consists of three domains: compute (includes storage), hypervisor, and infrastructure network

4.3 Compute Domain and Inter-Domain Interfaces

Common to all three NFVI domains is figure 2 [5]. It depicts the general domain architecture, and associated interfaces. Of particular interest to the compute domain is interface #11, interfacing the compute domain to the orchestration and management, and interface #14, interconnecting to the infrastructure network.

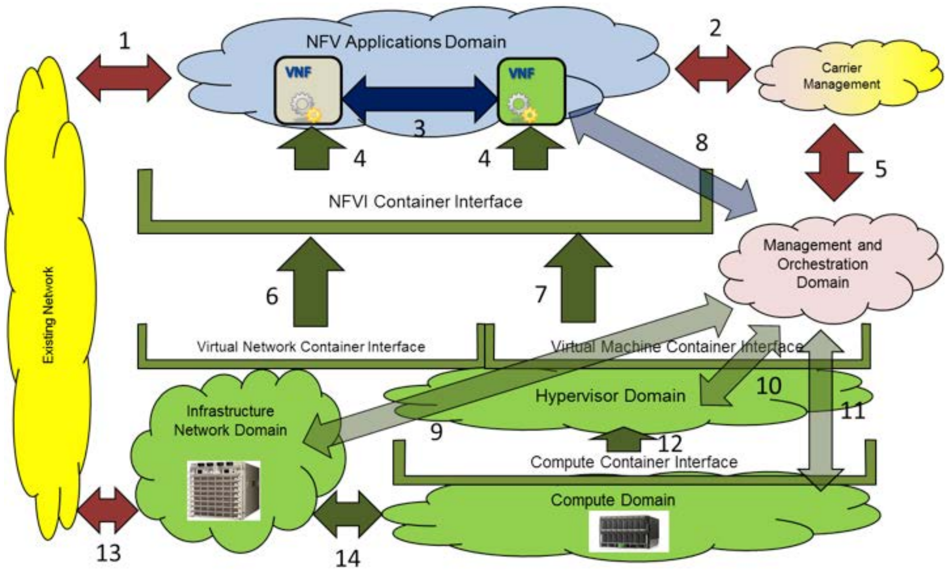
NFV Applications Domain
VNF
VNF
Carrier Management
Existing Network
NFVI Container Interface
Management and Orchestration Domain
Virtual Network Container Interface
Virtual Machine Container Interface
Hypervisor Domain
Infrastructure Network Domain
Compute Container Interface
Compute Domain
1
2
3
4
4
5
6
7
8
9
10
11
12
13
14

Figure 2: General domain architecture and associated interfaces

The basic functional blocks contained are shown in figure 2: processor/acceleration, network interface, and storage. All functional blocks are managed and orchestrated remotely through interface 11.

4.3.1 Relevant Interfaces

Table 2 depicts the relevant interfaces for compute domain.

Table 2: Relevant interfaces of compute domain

Relevant 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.
Infrastructure Interconnect Interfaces	8	**Virtual MANO Container Interface:** the interfaces 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.
Orchestration & Management Interface	9	Orchestration and management interface with the infrastructure network domain.
	11	Orchestration and management interface with the compute domain.
	14	Network interconnect between the compute equipment's and the infrastructure network equipment.
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.

4.4 Relation to NFV Architecture Framework

Figure 3 depicts the NFV reference architectural framework [5]. Of interest to the compute domain are:

- VNF-NFVI: this reference point presents the execution environment provided by NFVI to the VNF.
- VI-Ha: this is the reference point interfacing the virtualisation layer to the hardware resources, including compute and storage.

- Nf-Vi: this reference point is used to assign virtualised resources in response to resource allocation requests; forwarding and exchange state information.

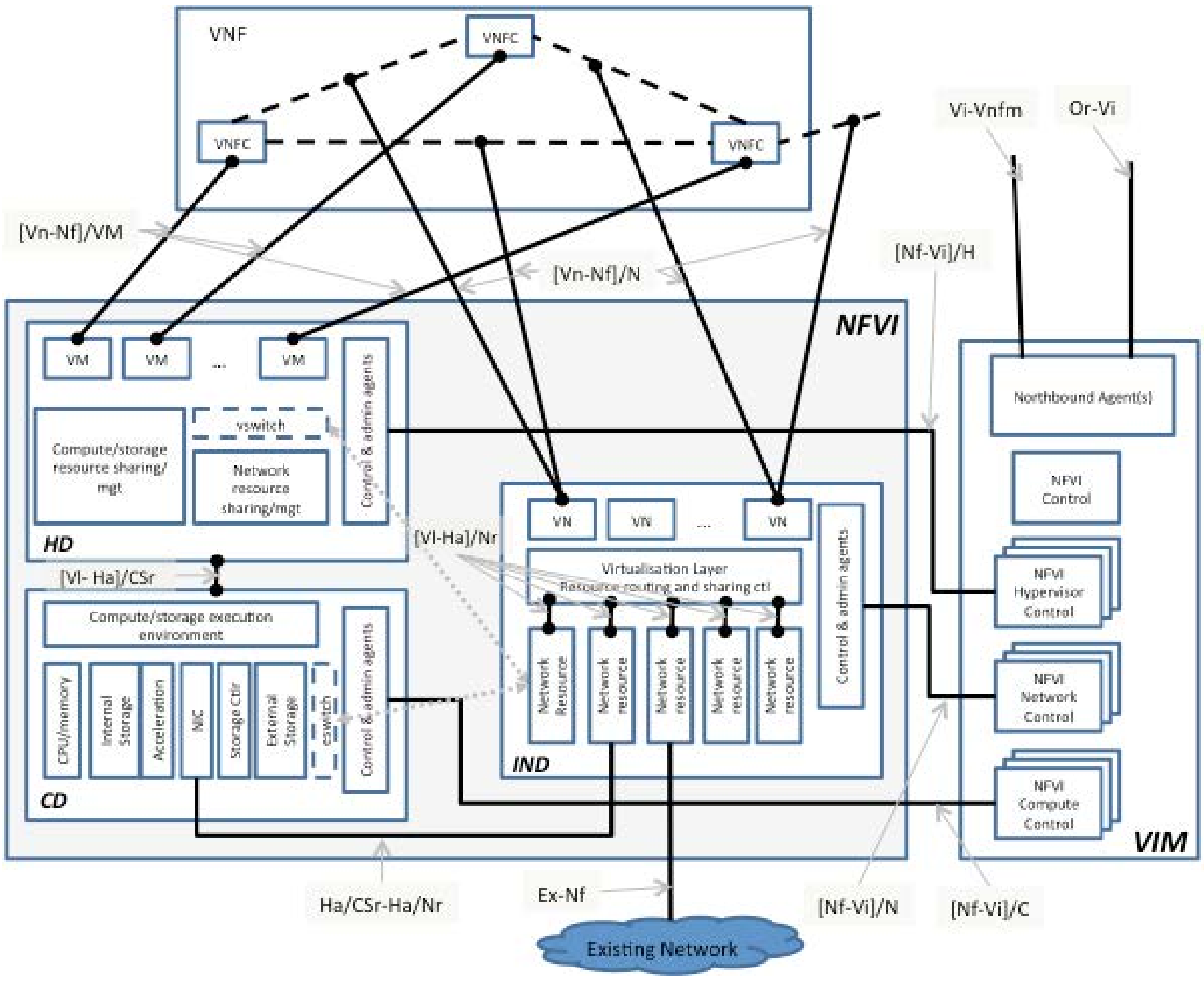

Figure 3: High-level overview of the NFVI domains and interfaces

Characteristics of NFVI reference points.

Table 3: Characteristics of NFVI reference points

INF Context	NFV Framework Reference Point	INF Reference Point	Reference Point Type	Correspondence with figure 2 Interfaces	Description and Comment
Internal	Vl-Ha	[Vl-Ha]/CSr	Execution Environment	12	The framework architecture [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 [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.
External	Nf-Vi	[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.

4.4.1 Elements of the Compute Domain

The compute domain consists of consists of a server, network interface controller (NIC), accelerator, storage, rack, and any associated components within the rack including the physical aspects of a networking switch and all other physical components within NFVI. For example a blade server including a built-in Ethernet switch - see figure 4.

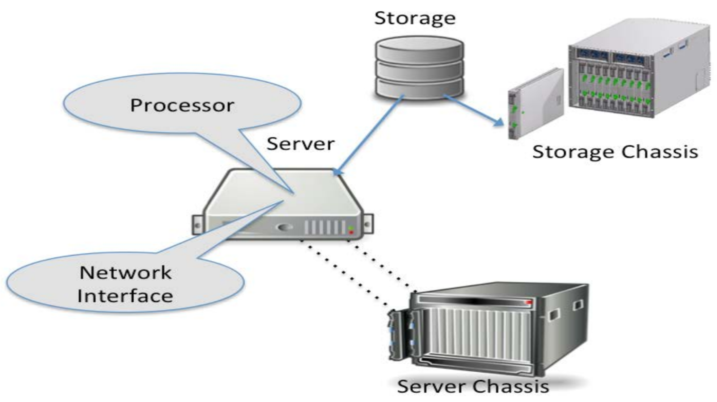

Figure 4: Functional elements of the Compute domain - Chassis can also be a Rack

4.4.1.1 Processor & Accelerator

General purpose compute architectures considered in the compute domain are ARM and x86. These two architectures are used as examples. They do not preclude the use of other instruction set architectures. In addition, accelerator functions for security, networking, and packet processing are also included.

4.4.1.2 Network Interfaces

The network interface could either be a Network Interface Card (NIC) which connects to the processor via PCIe or the network interface capability may be resident onboard the server. The NIC may optionally contain offload and/or acceeration functionality.

4.4.1.3 Storage

Storage addresses large-scale storage and non-volatile storage, such as hard disks and solid-state disk (SSD), including PCIe flash cards.

4.4.2 Influence of the Disaggregation Model

This clause discusses the form factor, in light of the Open Compute Project (OCP). In this form-factor, the CPU blades/chassis are separated from the NIC/Accelerator blades/chassis, and Storage blades/chassis. Inter-connection between the blades/chassis could be over optical fiber.

By separating critical components from one another, each resource can be upgraded or scaled independently. This increases lifespan for each resource and enables IT managers to replace just that resource instead of the entire system. There are thermal efficiency opportunities by allowing more optimal component placement within a rack, and by the relocation of power supplies from within the individual chassis to the actual rack.

By allowing for the disaggregation of hardware accelerators, OCP potentially provides for the sharing of accelerator resources across many computing or networking blades, thus providing improved amortization of traditionally expensive components.

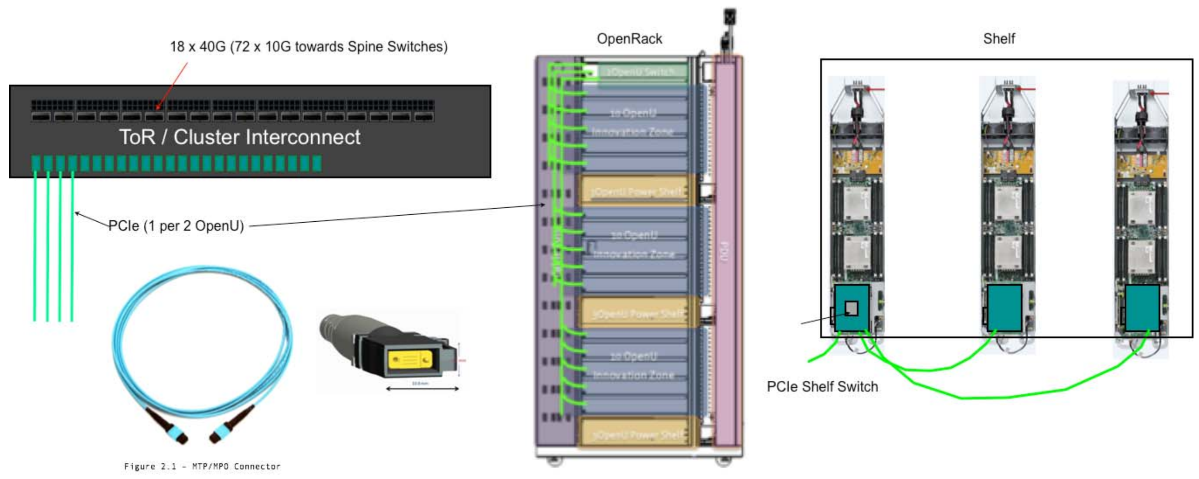
18 x 40G (72 x 10G towards Spine Switches)
ToR / Cluster Interconnect
PCIe (1 per 2 OpenU)
Figure 2.1 - MTP/MPO Connector
OpenRack
Shelf
PCIe Shelf Switch

Figure 5: Illustration of a typical Open Compute Project's Disaggregation model (Storage not included)

It should be noted that a single Compute Platform can support multiple virtual appliances, as seen in figure 6.

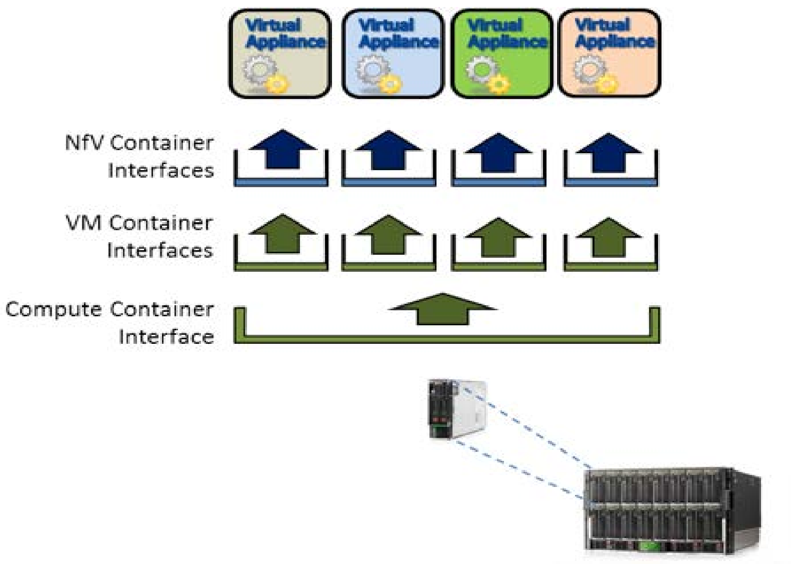

Figure 6: Compute platform can support multiple virtual appliances

4.5 Deployment Scenarios

The Compute Domain Infrastructure is the hardware on which VNFs execute. Since an NFVI-PoP is a location where VNFs execute, then an NFVI-PoP is a location where compute domain hardware is deployed. A building is a typical NFVI-PoP, especially for traditional equipment practice form factors. Compute Domain Infrastructure may be deployed in a variety of other locations and form factors from small pole mounted devices to shipping containers.

There are many elements with potentially separate security responsibility in different deployment scenarios: the building, the host compute hardware, the hypervisors and the guest virtual network functions within their virtual machines.

Below some deployment scenarios are described that are likely in realistic contractual arrangements.

For convenience they are summarized in table 4. The right-most column also identifies which NFV deployment scenarios are similar to the common deployment models used in cloud computing, as identified by NIST. It can be seen that the cloud computing industry uses deployment models that sometimes, but not always, relate to those expected for NFV. The NIST document also defines Cloud Service Models (Infrastructure, Platform and Software as a Service). It is possible that virtualisation of network functions might eventually become commoditized into a set of similar service models, but it is too early to be certain.

Table 4: Some realistic deployment scenarios

Deployment Scenario	Building	Host Hard-ware	Hyper-visor	Guest VNF	cf. NIST Cloud Model
Monolithic Operator	N	N	N	N	Private Cloud
Network Operator Hosting Virtual Network Operators	N	N	N	N, N1, N2	Hybrid Cloud
Hosted Network Operator	H	H	H	N	
Hosted Communications Providers	H	H	H	N1, N2, N3	Community Cloud
Hosted Communications and Application Providers	H	H	H	N1, N2, N3, P	Public Cloud
Managed Network Service on Customer Premises	C	N	N	N	
Managed Network Service on Customer Equipment	C	C	N	N	
NOTE: The different letters represent different companies or organisations, and are chosen to represent different roles, e.g. H = hosting provider, N = network operator, P = public, C = customer.					

4.5.1 Monolithic Operator

The same organisation that operates the virtualised network functions deploys and controls the hardware and hypervisors they run on and physically secures the premises in which they are located.

4.5.2 Network Operator Hosting Virtual Network Operators

Based on the 'Monolithic Operator' model, except as well as hosting itself, the network operator (e.g. BT, Verizon) hosts other virtual network operators (e.g. Virgin^{TM} Mobile, Tescos^{TM}, Walmart^{TM}) within the same facility. It would probably isolate each virtual operator on separate hardware. However, in theory, the virtual machines of different virtual network operators could run alongside each other over the same hypervisor.

4.5.3 Hosted Network Operator

An IT services organisation (e.g. HP^{TM}, Fujitsu^{TM}) operates the compute hardware, infrastructure network and hypervisors on which a separate network operator (e.g. BT^{TM}, Verizon^{TM}) runs virtualised network functions. The premises including cable chambers, patch panels etc. are physically secured by the IT services organisation.

4.5.4 Hosted Communications Providers

Similar to 'Hosted Network Operator', except the IT services organisation hosts multiple communications providers. Instead, the IT services organisation may host one (or many) wholesale network operators, which in turn host multiple virtual retail communications providers. In this latter case the IT services organisation would give controlled rights to run virtualised network functions to the wholesaler, which could in turn delegate rights to the virtualised retailers.

4.5.5 Hosted Communications and Application Providers

Similar to 'Hosted Communications Providers' except servers in a data centre facility are offered to the public for deploying virtualised applications (Cloud) while in the same physical facility network operators and communications providers deploy virtualised network functions on the same type of generic hardware platforms (probably not sharing the same server hardware, but in blades and racks alongside and sharing the same data centre network).

4.5.6 Managed Network Service on Customer Premises

A network operator runs virtualised network functions on its own generic server hardware located on a customer's premises and physically secured by the customer.

This model could be in a residential or enterprise scenario, for example, respectively, a remotely managed home gateway or remotely managed VPN gateways, firewalls, etc.

4.5.7 Managed Network Service on Customer Premises Equipment

Similar to 'Managed Network Service on Customer Premises', except the compute hardware is supplied and operated by the customer not the network operator. The customer allocates a blade to the network operator, which runs a hypervisor on this blade, and in turn runs all the necessary network functions within virtual machines over this hypervisor.

This model does not involve the customer running virtual machines on the same hypervisor and hardware as the network operator, although this would be another valid scenario (similar to 'Hosted Communications and Application Providers' except the customer both hosts the virtual network functions and runs guest applications in virtual machines alongside them).

In general, in order to determine the security implications of a deployment scenario, the two main factors are:

1) The different parties that operate each of the levels (building, host hardware, hypervisor, guest VNF).

2) Whether the party at any one layer has exclusive or non-exclusive use of the resources of the lower layers and, if non-exclusive, are the resources available to all other parties or only to a restricted set (e.g. the general public, or competing operators).

5 External Interfaces of the Domain

Given the modular nature of the compute domain and that it consists of server, storage and networking fabric, the various interfaces can be classified as:

1) Physical network interfaces.
2) Internal and external domain interfaces.
3) Management and orchestration interfaces.

5.1 Physical Network Interfaces

There are several different types of equipment interfaces that are likely to be utilized. While these might be the more popular interfaces, it is not intended to be an exhaustive list. These interfaces include:

- Ethernet.
- Fibre-channel.
- Infiniband.

5.1.1 Ethernet

Ethernet was introduced in 1980 and was standardized as IEEE 802.3 [i.2] in 1985. And it is the most pervasive of the technologies listed above. While an NFV environment may be deployed without Fibre-channel, Infiniband or NVMe technologies, it is doubtful that such an environment could be achieved without some type of Ethernet technology.

5.1.1.1 Nature of the Interface

The two most widely used interfaces for Ethernet include twisted pair (CAT 5/6) for up to 1GbE. And for 10GbE and faster, fiber optics is more of the norm through pluggable fiber optic transceivers known as SFPs (Small Form-Factor Pluggable).

While 10GbE deployments are primarily using fiber optics, today, the increasing availability of 10GBASE-T alternatives is helping to drive the adoption of twisted pair adoption due to the compatibility with slower 1GbE infrastructure.

5.1.1.2 Specifications in Current Widespread Issue

Today, the most widespread Ethernet specifications can be found in IEEE 802.3 [i.2].

5.1.1.3 Achieving Interoperability

Typically interoperability testing for Gb Ethernet has been provided by University of New Hampshire, conformity with IEEE standards and via industry "Plugfests".

5.1.2 Fibre-Channel

Fibre-channel (FC) is a high-speed networking technology that widely used for Storage Networking.

Fibre-channel utilizes an encapsulated form of the Small Computer System Interface (SCSI) command set over the Fibre-Channel Protocol (FCP) as a transport protocol.

5.1.2.1 Nature of the Interface

Fibre channel is available in the following speeds: 1, 2, 4, 8, 10, 16, and 20 Gbits/se However, only the 1, 2, 4, 8 and 16 interfaces are interoperable with one another. 10/20G FC has a different encoding scheme and is not compatible with the other technology.

Fibre channel devices can communicate directly with one another or they can be multiplexed together via a switch.

Fibre channel typically comes in a couple of cabling options such as LC (newer) or SC (older). And there are cables that can bridge the two different interfaces.

5.1.2.2 Specifications in Current Widespread Use

Currently, the largest install bases of FC are based on 4G and 8G.

5.1.2.3 Achieving Interoperability

Typically interoperability testing for Gb Ethernet has been provided by University of New Hampshire, conformity with T11 Technical Committee and via industry "Plugfests".

Additionally, for broadest interoperability, it is advised to use 1, 2, 4, 8 or 16G FC for backwards compatibility with legacy FC deployments.

5.1.3 Infiniband

Infiniband (IB) is a high-speed, low-latency switched fabric that is often used in the formation of high-performance computing clusters.

5.1.3.1 Nature of the Interface

Infiniband is a serial style of interface, sending a bit at a time, as opposed to 32 bits or 64 bits at a time. This serial bus can carry several channels of data at a time to achieve very high performance.

Furthermore, IB can support multiple memory areas, which can be accessed directly via processors as well as storage devices.

5.1.3.2 Specifications in Current Widespread Use

Currently, Infiniband Release 1.3 improves the testing of QDR and FDR speeds. And it also lays the groundwork for EDR, which is expected to emerge in 2013-2014.

5.1.3.3 Achieving Interoperability

Interoperability is typically accomplished by conformance with Infibiband Trade Association specifications as well as participation in the IBTA Compliance and Interoperability Plugfests.

5.2 Internal & External Domain Interfaces

There are two main interfaces that need to be discussed: One to the VIM and the second to the VNF application.

5.2.1 NFVI to VIM (Nf-Vi) Interface

There are two primary types of interfaces that the NFVI is concerned with. One is internal to the NFVI, such as the SWA-5 interface, which is used for resources, such as a virtual NIC, a virtual disk drive, virtual CPU, etc. It is not intended for use as a management interface, hence a VNF is not to use SWA-5 to manage the NFVI. This is primarily intended to logically fence off responsibilities, but is also intended for security reasons. Reasonable steps shall be taken to prevent unauthorized access, from within a VM, from attacking the underlying infrastructure and possibly shutting down the entire domain, including all other adjacent VMs.

The only authorized NFV management interface and interaction is via the Nf-Vi interface and that component that acts as the Virtual Infrastructure Manager (VIM). While the VIM can actually operate within a hosted VM, hence act as a VNF, its management interface is still via the Nf-Vi, though it might be abstracted as a SWA-5 interface. While this configuration is indeed very possible, it is not desirable, again due to security concerns. But there are also reliability concerns for not virtualizing the VIM on the same infrastructure that it is managing. In the event that the hypervisor domain of the NFVI requires a restart, the VIM will lose its ability to operate and continue to manage the NFVI. So, it should be an external service, process, etc.

5.2.2 Nf-Vi/C and [Vl-Ha]/CSr Interfaces

Within the NFVI, there are two interfaces external to the compute domain-not to be confused with the NFVI, which encompasses all of the infrastructure. These interfaces include the Nf-Vi/C, which is just a subset of that interface, which has already been described and which is used by the VIM to manage the compute and storage portion of the NFVI. And then there is the [Vl-Ha]/CSr interface, which is the interface between the compute domain and the hypervisor domain. It is primarily used by the hypervisor/OS to gain insight into the available physical resources of the compute domain.

5.2.2.1 Nature of the Interface

[Nf-Vi]/C: 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.

5.2.2.2 Specifications in Current Widespread Issue

The Nf-Vi interface, is most commonly executed over some type of Ethernet interface. However, management specifications are far and wide between. The DMTF CIM is one type of information model that is commonly used, across various elements of the NFVI such as for Compute, Storage and Networking. Likewise Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) are also commonly used. Storage infrastructure also makes wide use of SCSI Enclosure Services (SES). And Computing infrastructure may make use of out of band management via Intelligent Platform Management Interface (IPMI).

The point is that there are numerous types and models of management widely utilized today. While this may seem as though there are plenty of options to choose from, the reality is that there is not currently a consistent management specification across all of the NFVI components, which allows for both the getting and the setting of NFVI parameters, which is necessary for subsequent orchestration and portability.

5.2.2.3 Achieving Interoperability

Interoperability and compatibility are key components of the Portability requirement. In order to achieve this, there shall be a consistent set of definitions, semantics, functions, views and other aspects for interoperable management and orchestration across the NFVI. In order to achieve this, there is a need for a single, common representation or model, commonly referred to as an information model.

To date, much management capabilities are implemented via vendor unique APIs and tools. While this is expected to continue, there shall be an alternative non-vendor unique method for accomplishing the same purposes.

Also, there may be a need to examine how utilization statistics are obtained and presented in order to avoid generating excessive CPU utilization for obtaining, parsing and formatting of information. This too shall be performed in an economic and consistent fashion if there is to be interoperability.

5.3 Orchestration & Management Interfaces

Orchestration and management of the NFVI is strictly via the Nf-Vi interface. In order to properly work, there shall be an interface of some type. The type of physical interface is not as important as is the need for a dedicated and secure interface. It is this interface, which will be utilized to establish trust and compliance of the underlying infrastructure.

In addition to the interface, there shall also be relevant agents for each of the underlying domain components (i.e. compute, storage, hypervisor and infrastructure network).

5.3.1 Managing NFVI Hardware Elements

The NFVI hardware elements include:

- Rack.
- Fabric.
- TOR switch.
- Power grid.
- Rack shelf.
- Server chassis.
- Storage chassis.
- Accelerator blades/chassis.

Management of compute platforms particularly in a data center environment requires common models so that equipment from diverse sources can be managed together. One SDO with a suite of specifications for this purpose is the DMTF and its System Management Architecture for Server Hardware (SMASH). Its top level document, DSP0217 [6] SMASH Implementation Requirements has 78 normative references, mostly from DMTF. These include profiles for sensors, CPU, memory, fans, power supply, OS status, and many other components of servers. Together, they can be used to manage heterogeneous servers.

5.3.1.1 The Rack

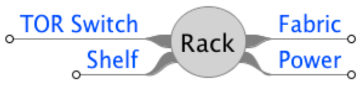

Figure 7

The rack includes the TOR switch connectivity, the fabric, shelf and power.

5.3.1.2 The Fabric

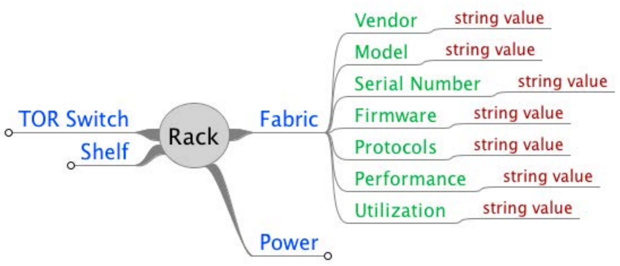

Figure 8

This is an optional component as not all racks utilize a fabric that is distinct from the TOR switch.

5.3.1.3 The Top of Rack (TOR) Switch

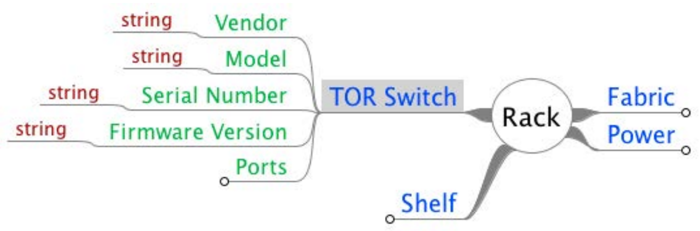

Figure 9

A TOR switch can be a discrete rack component, or it may actually be integrated into the backplane of a blade server that occupies a shelf within the rack.

5.3.1.4 The Power Grid

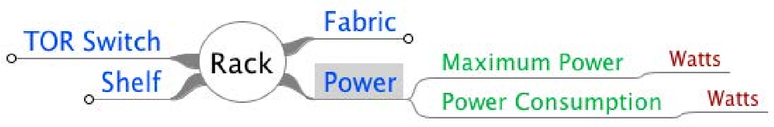

Figure 10

It is assumed that power of some nature will be supplied to a rack, if there are ever to be any sort of operations. This power may be single or multiple sources in nature. But the power may also be supplemented with a type of Uninterruptable Power Supply (UPS). This UPS, power strips and some other components may also have manageable interfaces, which can be used to provide power management information exchanges with the VIM.

5.3.1.5 The Rack Shelf Contents

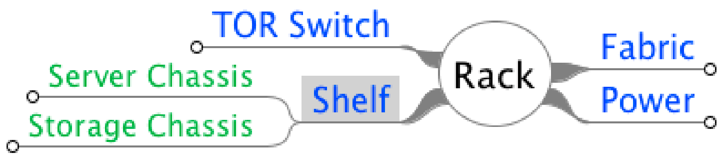

Figure 11

The rack shelf may contain a server chassis and a storage chassis.

5.3.1.6 Server Chassis

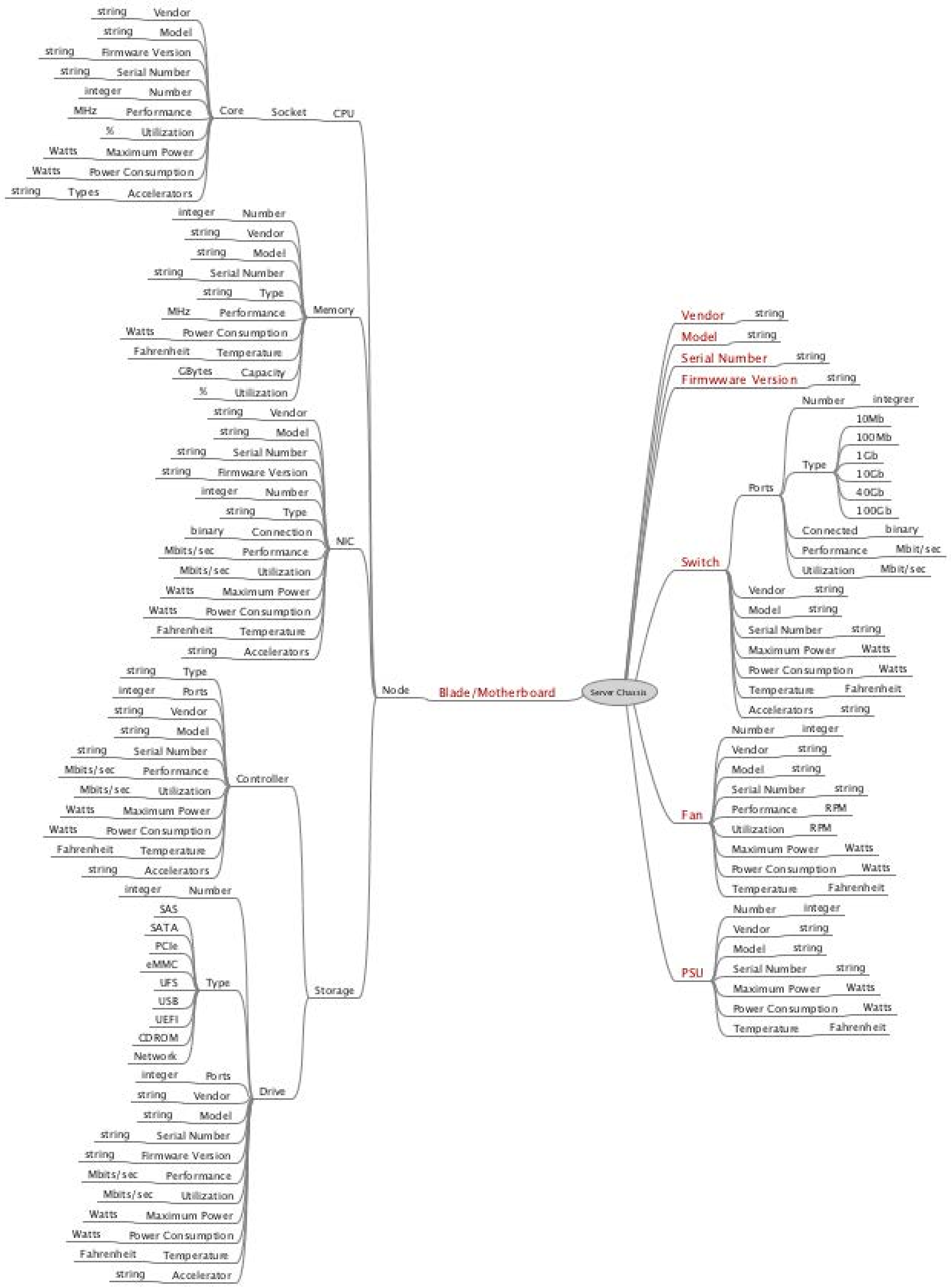

Figure 12

The serve chassis may contain all the element shown in the Mind Map above.

5.3.1.7 Storage Chassis

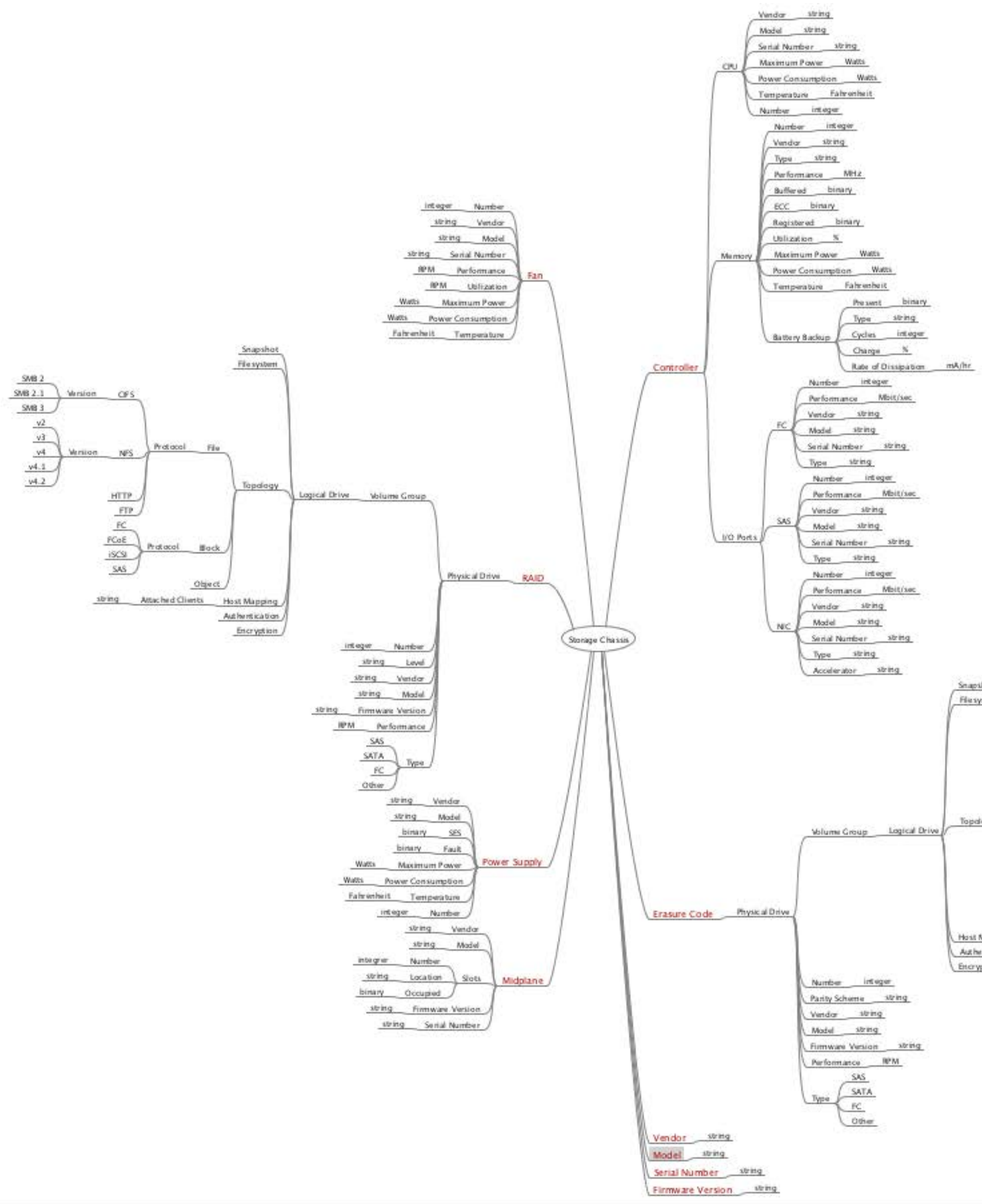

Figure 13

The storage chassis may contain all the element shown in the Mind Map above.

6 ISG E2E Requirements & Implications over Compute & Storage

This clause is dedicated to responding to the 3 ISG E2E documents ([1], [2], [4]) - All text in Italics affects the compute & storage requirements, and is taken from these E2E documents:

1) Architecture & Framework.

2) Requirements.

3) E2E use cases.

6.1 General (E2E requirements)

- [**Gen.1**] The NFV framework shall be able to permit Service Providers/Network Operators to partially or fully virtualise the network functions needed to create, deploy and operate the services they provide.

NOTE: Partial refers to virtualisation of a specific set of functions (e.g. based on similar characteristics) within a network system/subsystem, and used for creation of certain service. For example, virtualisation of network functions in the control plane but not the ones in the data plane.

- [**Gen.2**] In case of partial virtualisation, any impact on the performance or operation of non-virtualised network functions shall be manageable, predictable, and within the acceptable limits.

6.2 Performance (E2E requirements)

- **[Per.2]** The NFV framework shall be able to describe the underlying infrastructure requirements of a VNF so that it can be sized for a given performance target while the corresponding resources are allocated and isolated/shared in the infrastructure accordingly.
- [**Perf.3**] For any running VNF instance, the NFV framework shall be able to collect performance related information regarding the usage of compute, storage and networking resources by that VNF instance.
- [**Perf.4**] The NFV framework shall be able to collect performance related information concerning the resource usage at the infrastructure level (e.g. hypervisor, NIC, virtual switch).

Performance & Scalability (E2E Architecture & Framework).

- Performance and scalability are important since the implementation of a VNF may have a per-instance capacity that is less than that of a corresponding physical network function on dedicated hardware. Therefore, methods are needed to split the workload across many distributed/clustered VNF instances. Performance-efficient methods of deploying VNF instances on the NFV infrastructure can also be considered. It is necessary to study how to minimize performance degradation while keeping VNF instance portability on heterogeneous NFV infrastructure comprised of diverse virtualisation layer and hardware resources.

The compute domain shall provide metrics and statistics relating to the capacity, capability and utilization of hardware resources:

- CPU cores.
- Memory.
- IO (including accelerators).
- Storage subsystem.

In addition, mechanisms shall be provided for monitoring power usage.

The compute domain shall make the above metrics and statistics available to Virtualized Infrastructure Manager (VIM) through the Nf-Vi reference point. The VIM could further forward those metrics, if necessary, to the NFV Orchestrtor. Usage metrics represent current occupancy, but capacity and capability metrics maybe of use for planning purposes. To support Perf.4 requirement, for each network interface, the compute domain shall maintain statistics on:

- IO capability and usage per virtual interface (where supported).
- IO capability and usage per physical interface.
- Storage.
- Power (where supported).
- CPU Usage (Accelerator, where supported).

6.3 Elasticity (E2E Requirements)

- [**Elas.1**] The VNF vendor shall describe in an information model for each component capable of parallel operation the minimum and maximum range of such VM instances it can support as well as the required compute, packet IO, storage, memory and cache requirements for each VM.

- [**Elas.5**] The NFV framework shall provide the capability to move some or all VNF components from one compute resource onto a different compute resource while meeting the service continuity requirements for the VNF components.

NOTE: The timeframe required to move VNF components is dependent upon a number of factors, such as available network bandwidth and compute capacity.

6.4 Resiliency (E2E Requirements)

- [**Res.5**] The external storage instances/system shall provide a standard means for replication of state data (synchronous and asynchronous) and preservation of data integrity with the necessary performance to fulfill the SLAs.

See storage clause 7.3.

6.5 Security E2E Requirements

- Concern about Security of shared storage.

See storage clause 7.3.

6.6 Service Continuity (E2E Requirements)

Zero impact or seamless service continuity means that:

- in response to some anomaly (e.g. detected failure, commanded movement, and migration), there will be a means specified such that no observable state loss, no observable transmit queue packet loss and no observable transmit queue storage loss occurs; and

- any impact on latency and delay variation will be within the SLA specification for the function.

6.7 Service Assurance (E2E Requirements)

- [**SeA.1**] The NFV framework shall provide mechanisms for time-stamping of hardware (e.g. network interface cards, NICs, and network interface devices, NIDs, that sit beneath virtualisation infrastructure) . The minimum support from hardware shall be to:

  - copy packets or frames;

  - accurately time-stamp the copies, using a clock synchronized to a source of appropriate precision (e.g. IEEE 1588 [i.4]); and

  - forward the time-stamped copies to a configured destination. Once the precise time-stamps have been added in hardware, all other instrumentation and diagnosis functions can then proceed as virtualised functions without strict time constraints, e.g. filtering headers, removing payloads, local analysis, forwarding for remote analysis, logging, storage, etc.

- [**SeA.2**] It should be possible to interrogate whether particular network interface hardware provides hardware time-stamping facilities.Key Concept proposal - A NFV service is composed of different types (1 or more) resource pools. Each pool contains a Class of Service that has specific types of resources that are composed to deliver a specific subset of performance.

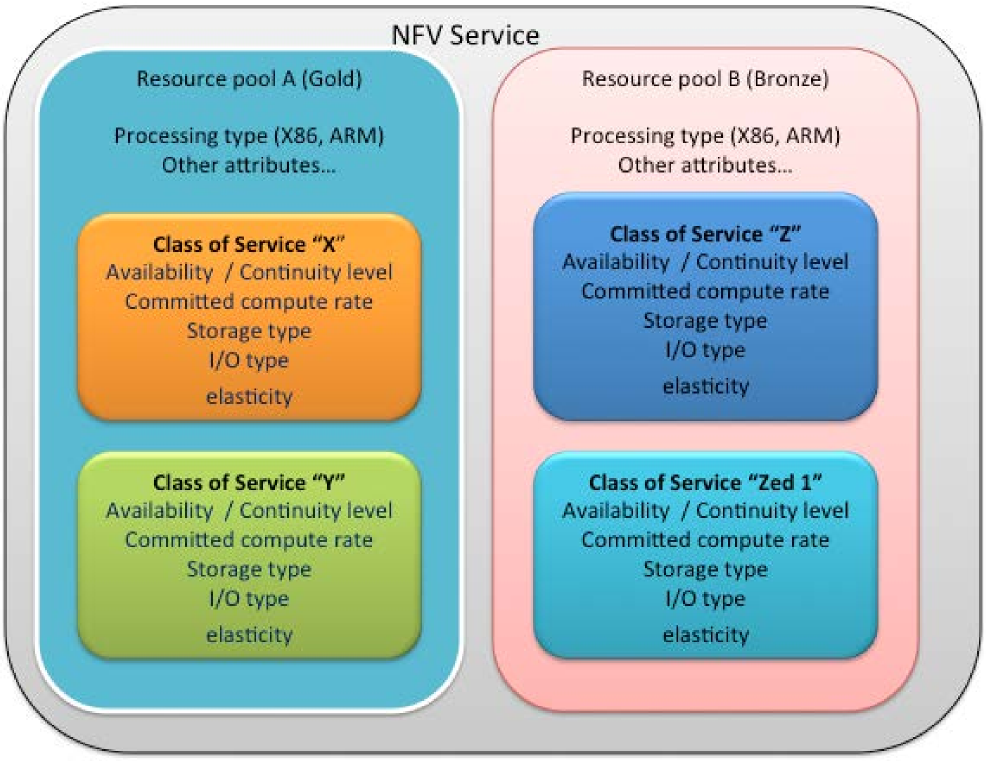

Figure 14: NFV types of services

6.8 Operations & Management (E2E Requirements)

- [**OaM.4**] As part of VNF life cycle management, the orchestration functionality shall be able to interact with other systems (when they exist) managing the NFV infrastructure comprised of compute/storage machines, network software/hardware and configurations and/or software on these devices.
- [**OaM.5**] The orchestration functionality shall be able to use standard information models that describes how to manage the VNF life cycle. Information models describe the deployment and operational attributes of VNFs, for example:
  - Deployment attributes and environment of a VNF e.g. VM images, required computational and storage resources and network reachability.

6.8.1 Statistics Collection - Objectives in NFV Framework

Statistics collection is important for supporting NFV management and orchestration functions, especially in resource allocation when instantiating VNFs and global resource management functions.

NFV-MANO performs resource allocation (compute, storage and network), as well as interconnection setup. A VNFD (virtual network function descriptor) describes the resource requirements of the VNF, and it is primarily used by the VNFM in the process of instantiating a VNF on the NFVI (NFV infrastructure), and for other life cycle management functions. The information provided in the VNFD is also used by the NFVO to manage and orchestrate network services, and virtualized resources across multiple VIMs. VNFD also contains connectivity, interface and KPIs (key performance indicators) requirements that can be used by NFV-MANO to establish appropriate virtual links within the NFV Infrastructure between VNF instances, or between a VNF instance and the legacy network. NFV-MANO keeps track of attributes of resource pools so that it can allocate the resources which can meet the VNFD described requirements, ideally in an optimal manner.

NFV global resource management performs resource discovery, resource optimization, access control, policy management, auditing, and fault management.

Statistics collection enables the hierarchy of management and orchestration functions to monitor that the resource pools (compute, storage and network) are functioning properly and are conforming to their specifications and expected performance and quality. In addition, statistics collection on VNFs provides additional insight into whether the VNFs perform as expected and whether relevant SLAs are adhered to. Some of the key performance indicators and key quality indicators which NFV-MANO uses are expected to be updated constantly by collected statistics. Examples of such indicators include average, maximum, and minimum values of throughput and latencies for certain VNFs or known functions, memory usage, power consumption of resources, number of active virtual machines running on a physical server, etc.

6.8.2 Proposed Hierarchy of resources and management software for statistics collection

The NFV management and orchestration architecture includes a hierarchy of resources, virtualisation layer, VNFs, and corresponding managers and management systems. The managers are responsible for collecting the relevant statistics.

Table 5 shows the resources and VNFs as well as the corresponding managers and the statistics collected and maintained.

Table 5: Proposed hierarchy of statistics collection

Resource/VNF	Manager(s)	MANO manager
Compute hardware	Hypervisor, virtualisation manager (e.g. OpenStack)	Virtualized infrastructure manager(s)
Storage hardware	Hypervisor, virtualisation manager (e.g. OpenStack)	Virtualized infrastructure manager(s)
Network hardware	Hypervisor and vSwitch (e.g. OpenStack)	Virtualized infrastructure manager(s)
VNF instance	EMS instance	VNF manager(s)
Service	OSS/BSS	Orchestrator

6.9 Energy Efficiency (E2E Requirements)

- [**EE.1**] The NFV framework shall support the capability to place only VNF components that can be moved or placed in a sleep state on a particular resource (compute, storage) so that resource can be placed into a power conserving state.

- [**EE.2**] The NFV framework shall be able to provide mechanisms to enable an authorized entity to control and optimize energy consumption on demand, by e.g. scheduling and placing VNF instances on specific resources, including hardware and/or hypervisors, placing unused resources in energy saving mode, and managing power states as needed.

- [**EE.3**] The NFV framework shall provide an information model that defines the timeframe required for a compute resource, hypervisor and/or VNFC (e.g. VM) to return to a normal operating mode after leaving a specific power-saving mode.

Energy Efficiency Considerations are outlined below:

Energy costs and environmental impact have become significant factors in the running of data centers and network infrastructure. As demands upon the network infrastructure varies, the number and configuration of VNFs varies to fulfil those demands. For management and configuration software to select optimal configurations requires information on the power consumption of configuring a particular NFVI to implement a VNF. The compute domain shall provide the necessary configuration and information functionality to support management and orchestration.

A number of standard interfaces have been developed to support the need to manage workloads across multiple servers to achieve the desired performance and power consumption. Notable established standards are:

- Intelligent Platform Management Interface (IPMI 2.0).

- Data Center Management Interface (DCMI 1.5).

- DMTF Systems Management Architecture for Server Hardware (SMASH).

Implementing standard interfaces to enable management of the NFVIs is desirable, and ideally should be consistent with Data Center management techniques. IPMI, DCMI and SMASH include features to support legacy servers, and do not preclude features for management of network infrastructure equipment. The Open Compute Platform organization is proposing a subset of these established standards for Open Hardware Management. Details are available on the Open Compute Platform website:

- Using existing standards appropriate to NFV i.e. functionality required to support the energy efficient implementation of VNFs minimizes the software support burden.

The requirements on the compute domain are:

- Req EE/C1: The compute domain shall provide use case specific power profiles to enable management and orchestration software to select the optimal configuration from the available NFVIs.
- Req EE/C2: The compute domain shall provide power management functions that enable the VIM to remotely control the power state of the domain.
- Req EE/C3: The compute domain shall provide information that enables the VIM to monitor the current power consumption of the domain.
- Req EE/C4: The compute domain shall provide information that enables the VIM to monitor the current temperature of components of the domain.
- Req EE/C5: The compute domain shall provide information that enables the VIM to predict the time it would take for the NFVI to transition from one power state to another.
- Req EE/C6: The compute domain shall provide a mechanism to limit the power consumption of the platform below a level specified by the VIM.

The compute domain shall be able to make autonomous power consumption decisions based on the current workload. This could be a mechanism local to the domain, as long as the VIM can monitor the temperature and control power, it can limit the temperature. The mechanisms used will modify the power profiles for a given use case. For example voltage and frequency scaling can be used to dynamically reduce the power and total energy consumed by the VNF. Knowledge of these capabilities would enable management and orchestration software to improve the configuration of NFVIs to achieve the required VNFs.

- Req EE/C7: The compute domain shall report the voltage scaling capabilities of the processor or processors.
- Req EE/C8: The compute domain shall report the frequency scaling capabilities of the processors or processors.
- Req EE/C9: The compute domain shall report the power state capabilities of the platform.

6.10 Additional points of interest (ISG documents)

- VNF life cycle implication:
  - Initiate.
  - Terminate.
  - Graceful shutdown.
- Compute & storage requirements:
  - Processing power, storage characteristics (e.g. size), Key Quality Indicator (KQI).
  - VM_Min/Max.
- Acceleration requirements:
  - Calculation intensive acceleration (e.g. encryption).
  - Data processing intensive acceleration (e.g. expression matching).

Req PER/C1: The compute domain shall support discovery of hardware (HW) /functional accelerators.

Req PER/C2: The compute domain shall support reservation of HW/functional accelerator resources.

Req PER/C3: The compute domain shall support binding/attaching to HW/functional accelerators.

Req PER/C4: The compute domain shall support unbinding/detaching from the HW/functional accelerator.

Req PER/C5: The compute domain shall support collection of the HW functional performance metrics.

7 Functional Blocks within the Domain

This clause describes the key functional blocks within the compute domain.

7.1 Definition of the "Compute Node"

An NFV compute node is a *coherent domain* (architectural entity) with an instruction set architecture that is managed as a single compute execution environment. A compute node includes a NIC (Network Interface Controller) to communicate with other compute nodes, network elements and storage elements.

In a COTS architecture the compute node is implemented as a multi-core processor and chip-set which may include the following physical resources:

1) CPU and chipset (instruction set processing element e.g. x86, ARM).

2) Memory subsystem.

3) NICs (certain classes of NICs include I/O acceleration technologies).

4) Optional hardware accelerators (e.g. a coprocessor).

5) Storage internal to the node (e.g. non-volatile memory, local disk storage).

6) BIOS/boot-loader (part of execution environment).

An optional Management Interface is a resource for control of the compute node as well as collecting statistics and alarms.

Metrics of the compute node are physical in nature (i.e. not processed by applying an algorithm) and exposed to the Hypervisor and Software Domains for consumption by VIM provisioning and management functions. A baseline set of these physical resource metrics is provided below in clause 7.4.

A COTS server blade is a typical example implementation of a Compute Node. The NFVI is required to continue to provide end-end service despite the existence of failures within compute nodes and the connectivity to those compute nodes (see ETSI GS NFV-REL 001 [i.3]). In practice, this requires the Compute Nodes to have multiple connections to independent (spatially diverse) network nodes within the NFVI Node. Figure 8 illustrates this aspect of Compute Node connectivity within the NFVI Node.

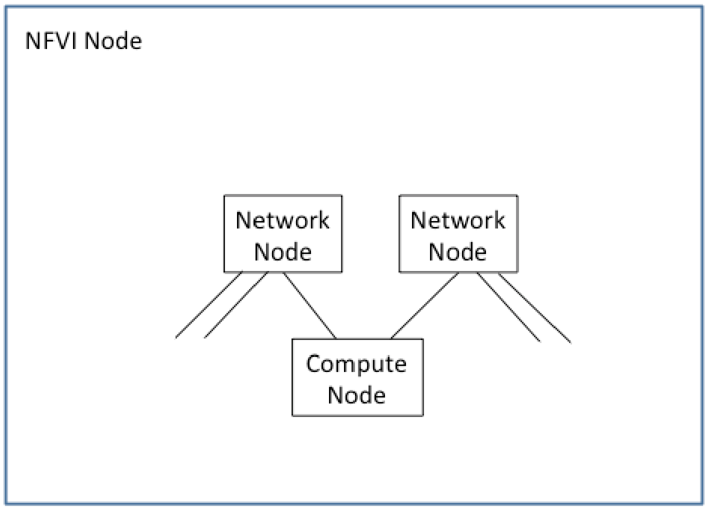

Figure 15: Compute Nodes shall have more than one network connection within the NFVI node

7.1.1 CPU Complex Overview

The CPU complex includes the processor and peripherals that support programming and processing of the network functions. The complex includes memory, I/O, processors and co-processors and interfaces.

From the NFV Network or an orchestration perspective, the CPU complex appears as a pool of processing resources that may be distributed across one or many nodes. From a single node perspective, the CPU complex appears as a set of limited number of processing elements, memory and I/O servicing the one node.

The CPU will need to scale and work with high memory capacity, storage sub-systems, co-processors, accelerators (if present), high I/O bandwidth, and processor interconnects (with one or multiple processor sockets on a single platform). The platform and interfaces shall be supported on industry standard high volume servers, and be compatible with open standards for switching, networking and storage.

The CPU complex will address the following: level of integration; hardware & software interfaces; hypervisor interfaces; instruction-set & programmability; number of CPU sockets; clock speed and form-factor.

7.2 Network Interface & Accelerators

This clause discusses the network interface to the CPU complex; I/O acceleration; and I/O features that support virtualisation.

7.2.1 Network Interface Controller (NIC)

The main function of the NIC is to deliver network I/O (physical or logical) to/from the CPU.

NICs may include hardware acceleration engines for encryption, encapsulation, forwarding and switching.

7.2.2 Accelerators

NFV Infrastructure (NFVI) is composed mainly of standard IT servers; network elements, such as routers, switches, and firewalls; and networked storage. However, some VNFs may require some form of hardware acceleration to be provided by the NFVI to meet their performance goals. While the standard IT based NFV architecture generally addresses scalability well, some use cases are more challenging, especially relating to VNFs that shall meet certain latency or SLA requirements. Various solutions and proposals are currently made for NFV Hardware Acceleration (HWA), all based on varying architectures.

The vision and objective of NFV is to implement network functions in portable code which can be executed on generic compute node infrastructure. However, in practical reality, 'generic' does not mean completely identical and even in the cloud computing environment, there are options and differences in the detailed specification of compute nodes. In the context of NFV, these options and differences often affect the packet throughput performance and therefore there is advantage in exploiting optional features if they are available.

These optional features where differences in compute nodes may arise include:

- Hardware accelerators such as encryption, digital signal processing (DSP), packet header processing, packet buffering and scheduling etc. which are either in a separate chip or integrated onto the same chip die as the CPU - system on a chip (SoC) - and either have direct access to the memory bus or have access to an on-chip system bus (e.g. PCIe).
- Additions to the Instruction Set Architecture (ISA) (e.g. x86, ARMv8, etc.) which implement new software acceleration features.
- Optional additions to the ISA for cache management operations where the behaviour of the instructions may be different on different chips of the same ISA.

However, exploiting these optional features is likely to compromise the portability of VNFC code unless a hardware abstraction layer is introduced. The role of a hardware abstraction layer is to present a single API independent of the hardware options such that the code of the VNFC can be written to use this API and thus be portable. All code which is dependent on the hardware options is within the hardware abstraction layer. Over time, CPUs add new instructions to accommodate accelerator functions. Therefore, the acceleration API needs to enable new CPU accelerator functions - see figure 9.

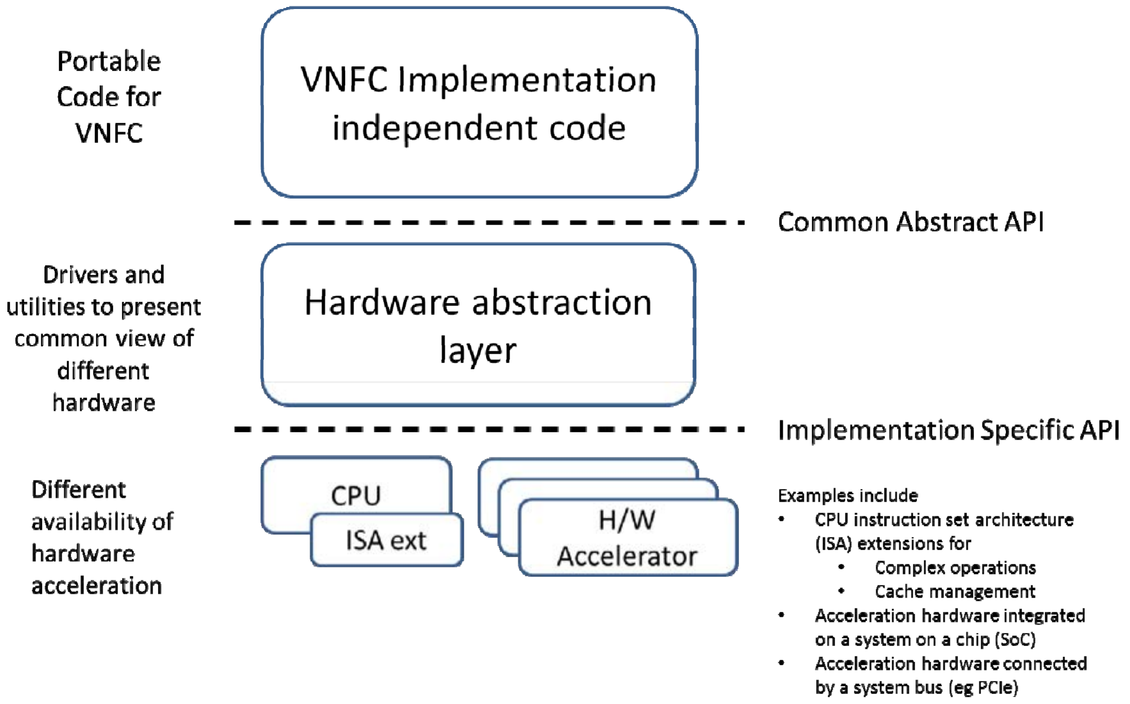
Portable Code for VNFC
VNFC Implementation independent code
Common Abstract API
Drivers and utilities to present common view of different hardware
Hardware abstraction layer
Implementation Specific API
Different availability of hardware acceleration
CPU
ISA ext
H/W Accelerator
Examples include
• CPU instruction set architecture (ISA) extensions for
• Complex operations
• Cache management
• Acceleration hardware integrated on a system on a chip (SoC)
• Acceleration hardware connected by a system bus (eg PCIe)

Figure 16: Use of a hardware abstraction layer to enable fully portable VNFC code across servers with different options and levels of hardware acceleration

The abstraction of accelerated functions benefits the programmability, portability and scalability of VNFs.

For portability, the abstraction API needs to enable moving of the VNF yet have different ways of implementing accelerated functions. The implementation of such functions may vary between vendors.

For scalability, an abstraction needs to be at the right level, so that innovation can occur below the abstraction. The abstraction needs to be efficient, hence there is a need to be pragmatic about the level of abstraction. The right balance will allow innovation but maintain efficiency of the API. An example that demonstrates these goals are feasible is the OpenGL API from the graphics world.

This hardware abstraction layer is a normal feature of operating systems and is normally implemented using a common driver model and hardware specific drivers. In the ETSI NFV architecture, there is a hypervisor which is the forms the base operating system which interfaces to the hardware. The hypervisor then provides common and uniform virtual hardware to all virtual machines so that a VM image is fully portable.

Therefore, in the target NFV architecture, in order to maintain this same level of portability, the hardware abstraction layer should be entirely contained in the hypervisor. In this way, the all virtualised hardware presented to the VM is generic so that the host operating system of the VNFC can use generic drivers - see figure10. Note that the Common Abstract API in figure 9 is realized as the Abstract Hypervisor API in figure 10.

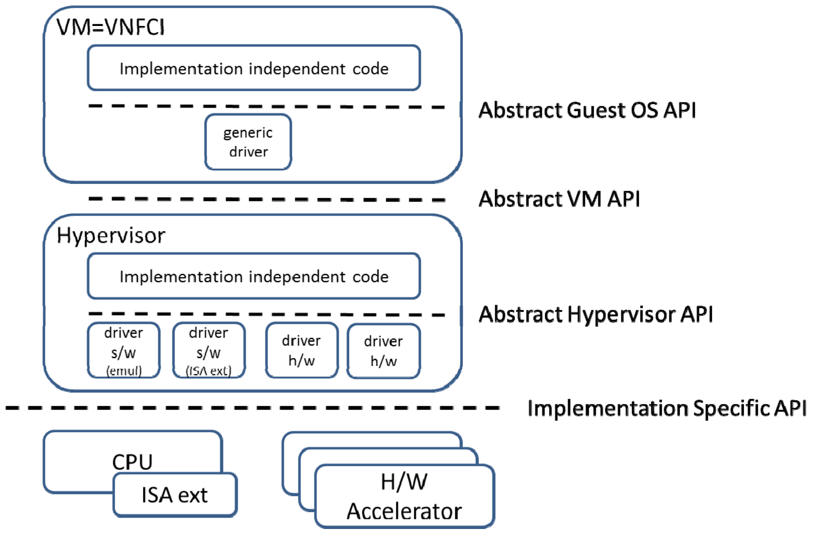

Figure 17: Target hardware abstraction layer implemented entirely within the hypervisor

However, current implementations do not always follow the target architecture of figure 10. In many current implementations, where hypervisors do provide full hardware abstraction and offer full portability of VMs, this often compromises the packet throughput performance. As a way around this this, many hypervisors enable improved performance by directly passing-though communication between a guest OS and acceleration hardware. The hypervisor does not attempt to abstract the communication to a common API. While this many enable improved packet throughput performance, this compromises the portability of the VNFC code of the VM image.

An intermediate, interim architecture, with a reduced level of portability can be achieved by creating the abstract API within the guest OS of the VNFC. In this intermediate architecture, the guest OS of the VNFC shall carry sufficient drivers for the range of hardware options. It is probable that different drivers are necessary by hypervisor type and by hardware option (figure 11).

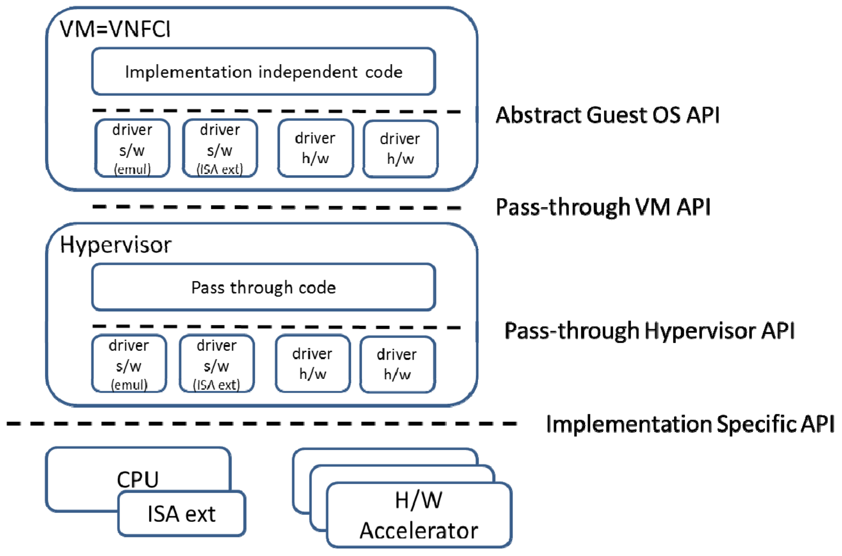

Figure 18: Interim architecture with hardware abstraction layer implemented by the hypervisor and the guest OS

Below, there are two types deal with two types of acceleration examples: Hardware and heterogeneous.

7.2.3 Hardware Acceleration

Hardware acceleration is the use of specialized hardware to perform some function faster than is possible by executing the same function on a general-purpose CPU or on a traditional networking (or other I/O) device (e.g. NIC, switch, storage controller, etc).

These functions may be correlated to the three NFVI domains and subsequently address Compute, Network and Storage Acceleration. By using the term "functions", one can abstract the actual physical implementation of the accelerator, which could be done by specialized hardware, software, firmware or any combination thereof.

This clause covers the options for ASICs, network processors, flow processors, FPGAs, multi-core processors, etc. to offload the main CPU, and to accelerate workload performance.

7.2.3.1 Need for Hardware Acceleration (HWA) Abstraction

To allow multiple HWAs to co-exist within the same NFVI, and to be used by various applications (VNFs), a unified approach and standard interfaces are proposed. Moreover, this mechanism will allow sharing of HWA technologies for acceleration based on various metrics, such as resource availability and/or utilization. Otherwise, these applications will not meet the portability requirements in NFV. This is due to the nature in which the underlying applications being accelerated depend on and use application programmable interfaces (APIs). Without a common HWA abstraction layer (aka, HAL), these applications shall be hard coded to the specific API utilized by the associated HWA.

These abstractions should be fairly general in design and should be representative of functionality. For example, a class of accelerator types that deal with compression could be created such that an application that requires the use of a compression operation can detect whether there are hardware acceleration capabilities present and then forward such functions to the appropriate accelerator.

If no HWA is present, then the application may be able to complete its function through a provided library, or possibly even through an emulated HWA, or some other means. The intention of this abstraction is to ensure the application (VNF) and the underlying HWA is decoupled so that either the specialized hardware can be absent or that it can vary from one manufacturer to another without requiring new VNF code. It is understood and acceptable to address the different hardware options through a device driver, virtual machine, QEMU emulation or similar means within the NFVI's host operating system or hypervisor.

7.2.3.2 Need for Dynamic Allocation and Management

In order to accommodate the possibility of a disaggregated deployment (i.e. a deployment where the HWA is physically decoupled from a specific host, and shared between two or more NFVI compute or networking resources), some form of management will be needed in order to ensure higher-level orchestration capabilities, utilization monitoring, and even customer SLAs are supported.

This management support should take into consideration that HWAs may be deployed as a pool in which resources are dynamically allocated to other specific resources within the NFVI or to specific VNFs based on needs and policies, and that such assignments will need to be recorded, modified, suspended, or possibly even support live migration.

7.2.3.3 Hardware Accelerator Example

Next-Generation Fire Wall (NGFW) is a good example of a network service that could benefit from the use of hardware acceleration, in an accelerated NFVI server architecture.

Potential NGFW workload may consist of the following.

Potential data plane functions using acceleration (e.g. on the NIC/Accelerator):

- L2 forwarding and virtual bridges.
- L3 routing and virtual routing.
- IP VPN termination.
- Network Address and Port Translation (NAPT) - multiple modes of operation.
- Statefull network flow analysis.
- Packet classification and filtering.
- Dynamic per flow application policy.
- Load balancing to x86 or egress interfaces.
- SSL (Secure Socket Layer) identification.
- SSL inspection.

Application plane on host CPU:

- DPI for application and protocol identification.
- Snort for threat management.
- Passive network discovery.
- Targeted vulnerability assessment.

Control plane on host CPU (x86 or ARM):

- ARP (Address Resolution Protocol).
- OSPF (Open Shortest Path First).
- RIP (Routing Information Protocol).
- Application management GUI (Graphical User Interface).

Above functional partitioning is workload-specific and will depend on the IPS/Firewall application. Performance can be CPU-bound, I/O bound or both. Specific requirements will impact the underlying implementation.

7.2.4 Heterogeneous Acceleration

Heterogeneous accelerators are another class of accelerated functions called from the VNFCi. It refers to functions implemented within the compute node on the NIC, CPU Complex, accelerator blades/chassis, a PCIe plug-in card or a PCIe attached device such as FPGA, ASIC, NPU, and called from the VNFCi, possibly on a fine granularity.

Heterogeneous acceleration techniques may be independent of, or may rely on the CPU Complex and NIC hardware features. Software may make use of techniques such as huge page memory, ring buffers and poll-mode drivers.

Implementation of heterogeneous accelerators may vary from vendor to vendor.

7.2.4.1 CPU Complex, NIC, and PCIe-attached Devices Accelerator Examples

In a typical packet processing architecture, Packets enter via ingress interfaces and are then classified into a number of abstract queues. A scheduler distributes queued packets to software threads, or other accelerators (for example Crypto). Accelerators may be physically located on a NIC, PCIe Plug-in card or CPU complex, the CPU complex and NIC may also be combined onto the same SoC.

The list below identifies accelerators that may benefit data-plane applications.

7.2.4.2 Examples for functional accelerations

- Classification accelerator.
- Crypto.
- DPI/Regular Expression.
- Compression/Decompression.
- Buffer manager.
- Queue manager.
- Work scheduler.
- Timer manager.
- Traffic manager.
- Address translation.

7.2.4.3 Accelerator Abstraction API Examples

One well known model for abstracting heterogeneous accelerators is using abstract queues and processing elements. Abstract queues hold work (for example: packets, timers and messages) that are waiting for further processing.

Processing elements process work dispatched from the abstract queues. Processing elements include both programmable CPU's and fixed function accelerators.

In the queues and processing elements model, a scheduler selects work from the abstract queues and distributes the work to processing elements according to a specified scheduling algorithm. Processing elements produce output results and post the results to other abstract queues. In a networking system, network interfaces act as the ultimate sources and sinks of work (packets).

The classifier, queue and potentially the scheduler are accelerators that are abstracted, or the equivalent functionality is implemented in optimized software.

7.3 Storage

This clause addresses large-scale storage and non-volatile storage, such as hard disks and solid-state disk (SSD), including PCIe flash cards.

Storage hierarchy in the compute can consist of, but is not limited to the following:

- Cache Storage.
- Primary Storage.
- Secondary Storage.
- Cold Storage or Archived Storage.

Each of these are characterized by different levels of latency, costs, security, resiliency and feature support.

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. These caches can also be automated to form a tiering function for the storage infrastructure.

7.3.1 Storage Building Blocks

The building block components for these various levels or types of storage infrastructure can include four primary classifications of drives: Hard Disk Drives (HDDs), Solid State Disks (SDDs), Optical Disks, and Tape Drives.

In addition to the drives, storage building blocks can include data protection, compression, de-duplication, protocol or access controllers, backplanes, carriers, power supplies and enclosures.

7.3.1.1 Hard Disk Drives

Hard Disk Drives (HDDs) typically consist of one to several platters and an actuator with a magnetic recording head. The platters are typically coated with a magnetic substrate that can receive a charge from the recording head that is used to represent data. Currently, there are many benefits to the use of HDDs as storage components. Their sequential performance can be relatively good in comparison to other technologies. And with the exception of tape-based media, they currently offer the lowest media costs.

However, HDDs can become considerably slow when the head movement actuates from inner to outer disk. This can be exacerbated when several layers of abstraction are placed between underlying physical disks and the upper layer application.

7.3.1.2 Solid State Disks

Solid State Disks (SSDs), unlike HDDs, do not possess moving parts, hence there is no notion of inner vs. outer disk to be adversely affected by any actuation. As such, they are a suitable technology for use in applications where there would otherwise be considerable randomized accesses.

Despite the rapid price drops that SSDs currently face, they are relatively much more expensive and smaller in capacity than their HDD counterparts. This means that usage within NFV Infrastructure shall be carefully considered to avoid excessive costs.

7.3.1.3 Cache Storage

Increasingly, the compromise between the cost effectiveness of Hard Drives and the performance efficiency of Solid State Disks is making its way into data centers in the form of hybrid caching storage. These storage arrays could be utilized as block-based storage or as file-based storage. What makes them hybrids is that they will utilize SSDs as a high-speed cache (or possibly as a tier) and will utilize HDDs for persistent storage, such as backend RAID.

While there are many solutions emerging with such caching capabilities, additional evolution of this type of storage adds policy based management to this caching layer so that some data, which might be more critical or frequently accessed, might remain within the cache as opposed to simply being flushed after some time interval is reached.

In order to appear seamless to the clients services that might be connected, these caching layers effectively map the blocks between at least two different sets of devices. The higher speed device, which is assumed to be the cache, is typically smaller in range than the slower device, which should be comprised of a larger range of blocks. There is no hard and fast rule with respect to the ratio of blocks. Suffice it to say, it is really a matter of budgets versus performance requirements.

7.3.1.4 Cold Storage

Cold Storage, or archive storage, is storage this is used for data that needs to be maintained for historical purposes. While implementations can vary based on the use low cost SATA drives to LTO tape, the goal of a cold storage system should be to free up performance-based storage for more important tasks. With that said, it should also be understood that one of the goals for NFV is to reduce the energy consumption of data center deployments. And that suggests a look at technologies that can best accomplish that purpose.

With the introduction of LTFS (Linear Tape File System) in 2009, LTO tape has experienced a resurgence in the market as a viable storage medium, but not just for backup. LTFS allows an LTO-based library to be utilized as file store, but without requiring backup software and staging areas for accessed to archived data. Instead, the tape can be mounted in the same way as a disk drive is mounted today.

LTO also has an advantage over any other storage technology in that it is inherently designed to resist bit flip, which makes it suitable for very long archival periods-beyond 10 years. It is also requires considerably less power to operate than a disk drive and is designed to be powered down when not active, which makes it the lowest consumer of power of any available storage technology, today.

7.3.2 Storage Topologies

Storage comes in different topologies. It can be Direct Attached Storage (DAS), Network Attached Storage (NAS), or even Storage Area Networked (SAN). Each choice carries with it certain pros and cons. But there are other considerations that shall be made when it comes to interactions and support with a hypervisor, guest operating systems, within the hypervisor domain, and even in relation to certain high resiliency practices or features:

1) Direct Attached Storage (DAS), comes in various forms. While typically associated with internal server hard drives, a better way of thinking about DAS is that it is captive to the server to which it is attached.

2) Storage area networking, a disk block based storage technology, is probably the most pervasive form of storage for very large data centers and has been a de facto staple as it relates to database intensive applications. These applications require shareable storage, large bandwidth and support for the distances from rack to rack within the datacenter. Ethernet, Fibre Channel (FC), and Infiniband (IB) have been the interface of choice as it was able to meet these requirements, primarily bandwidth.

3) Network attached storage, while typically associated as the de facto storage for unstructured data (file data vs database data), continues to grow in market share, largely due to the growing popularity of Virtualisation.

7.3.2.1 Block-based Storage

Block based storage, such as FC, iSCSI, or SAS, whether direct attached or attached as part of a shared SAN, has many benefits in terms of performance and snapshot capabilities. These snapshot capabilities, especially when used as part of Logical Volume Manager (LVM) has the ability to provide derivative snapshot capabilities, which could be used for cloning volumes.

A derivative snapshot, in this case, is a 100 % snapshot volume where a the snapshot has the ability to break the Copy-On-Write (COW) chain and to be used as a new master volume versus being using for restoring its own master volume to a prior state. The reason is a 100 % snapshot is it shall contain an entire chain as opposed to only containing some differential blocks.

While derivative snapshots can be quite useful for a virtualisation environment's image store, it is not necessarily the most widely utilized topology for large data centers due to complexities with provisioning and due to the fundamental lack of scale-out capabilities, though this will be addressed under the discussion about scale-out-storage, specifically with use of parallel NFS (aka pNFS or NFS v4.1).

Additionally, block storage can be relatively easily incorporated into various levels of resiliency schemes such as asynchronous or synchronous mirrors. Typically, SAN controllers will utilize a copy-on-write mechanism to keep the local copy and the mirror volume synchronously mirrored, though this would typically be on a LAN vs. on a WAN connection. The synchronous nature, when combined with a WAN, could result in unacceptable latencies as local writes will not be acknowledged until the remote writes are first completed.

Image stores probably do not require considerable scale since their capacity should be limited to OS and to applications. And performance impact should be relegated to boot storms, unless swap partitions are also utilized in pure HDD deployments. Swap partitions or files should not be utilized in combination with SSDs, as this will significantly deteriorate the life of such a drive due to the excessive writes.

Usage of block storage does bring certain complications at the provision process due to the need to create a LUN and to map or assign a client's initiator to that LUN. Additionally, some level of authentication may also be required.

7.3.2.2 File-based Storage

File-based storage systems are typically synonymous with Network Attached Storage (NAS) and consist of a storage array, some type of controller and operating system and one to several networked-storage protocols such as CIFS, NFS, iSCSI, FC, AoE, or something else.

One of the advantages with file based storage is the ability to treat a file as a block device or disk drive. Files can be easily appended to in order to create larger virtual drives. And files can be easily replicated to other locations.

One of the disadvantages of working with file-based storage is the inability, at least currently, to rapidly clone an existing virtual disk image into a new image. Currently, a byte for byte copy can take a considerable amount of time, based on the size of the disk image and the speed of the underlying physical storage, whereas a snapshot clone of a volume using LVM can take approximately one second.

Another potential complication of working file-based storage is the inability to establish synchronous and asynchronous mirrors without creating such mirrors at the physical hardware level. While there are some good commercial solutions for this, there are not currently any mature open source ones apart from scale-out solutions.

Today, Network File System (NFS) stands as the most widely deployed storage topology for large-scale virtualisation environments. This is largely due to the ability to quickly provision a pool of storage, utilize snapshots and backup at multiple levels in the system and to easily provision security policies for users and for services.

7.3.2.3 Object-based Storage

Over the years, there have been many forms of object based storage devices, services and products, though they have not always been successful at large scale adoption versus other NAS or SAN technologies. This is largely due to them being somewhat misunderstood from a features and benefits perspective. With that said, one of the key advantages of object storage is the ability to directly couple unique methods or security implementations with the actual data as opposed to having such capabilities come from an adjacent system or service. This enables storage to be more closely coupled in features to the specific needs of a service. A prime example of such a commercial usage is Amazon's Simple Storage Service (S3), and an example of an open source solution is the Swift plug-in within OpenStack.

7.3.2.4 Scale-out Storage

Scale-out Storage has many of the attributes of file-based storage, but is often distributed across several to many boxes for increased performance, resiliency or for both. There are numerous commercial solutions available, today, as well as quite a few open source ones. While the characteristics of the various solutions may seem similar at a high level, there are certain key differentiators that each solution typically strives towards:

- Striping-typically for HPC environments where redundancy or long term persistence of data is not as important as the overall performance.
- Mirroring-these solutions typically rely on various types of mirroring to replicate data in 2 or locations to protect against failures, but tend to be considerably slower than striped solutions, and in some cases it can take as many as 5 more nodes to surpass a single node's performance.
- Parity-based striping-provides a balance between the two other types, but is generally part of commercial solutions such as EMC's Isilon solution.

In 2010, the NFS specification was ratified to include v4.1 (aka pNFS) which adapts previously utilized NFS environments with scale-out capabilities. To date, NFS v4.1 is the only scale out solution to be managed under an SDO.

pNFS separates data and metadata to prevent the metadata server from becoming a bottleneck. Instead, it passes a layout driver to a pNFS-aware client that contains a layout driver. This layout driver tells the client how to directly access the desired data to avoid a common choke point.

Today, the open source layout driver supports autonomous I/O nodes, mirror nodes and striped nodes, but not parity-striped nodes to balance higher performance with resiliency.

Also, pNFS can utilize NFS, FC, iSCSI or Object Storage servers as its I/O server nodes.

7.3.3 Serviceability

7.3.3.1 Live Migration

Live migration, also known as vMotion in VMware, is a technology that is also available with the Kernel Virtual Machine (KVM) and with Xen-two Linux-based hypervisors. Live Migration is more than just failover support of virtual machines. It allows for the maintenance and support of the underlying infrastructure without requiring the guest virtual machines to be powered down. In short, Virtual Machines (VMs) can be migrated from one host machine's hypervisor to another host machine's hypervisor seamlessly and without any interruption to any clients' interactions with features provided by the initial VM. For any Software as a Service (SaaS), this feature is an absolute requirement.

7.3.3.1.1 Local Storage

This clause will discuss local storage requirements in regards to capacity, access time and latency. Local storage can typically consist of the following types of storage:

1) SATA (Serial ATA).

2) SAS (Serial Attached SCSI).

3) SSD (Solid State Disk).

There are benefits of each of these types of storage. SATA disks are the least expensive. SAS drives are typically more robust due to higher rotational speeds than SATA disks. However, SSDs continue to grow in popularity due to their benefits of having zero rotational latency. But they are the most expensive of the three technologies and can be difficult to justify due to such significantly higher costs.

7.3.3.1.2 Remote Storage

This clause will discuss remote storage requirements in regards to capacity, access time and latency.

Remote storage has two dominant types, though this is not all inclusive. The two preferred types of storage include NFS (Network File System) and Fibre-Channel (FC).

NFS has grown in recent years as the primary storage type for shared virtualisation storage and this is largely due to the ease of provisioning.

FC storage is also quite popular due to the shared, high-availability nature of FC-based SANs. However, the added steps of provisioning LUNs makes this specific type of block storage largely used in database oriented services as opposed to others.

Other interfaces which may be utilized include iSCSI, SAS-based SANs and even CIFS (Common Internet File System).

It should be noted that as the popularity and support of the open source Cloud project known as OpenStack, a new class of object storage based on the Swift plug-in may begin surfacing in the near future. Object storage provides numerous benefits over block or file storage. However, the difficulty of deployment is likely to be high.

7.3.4 Management

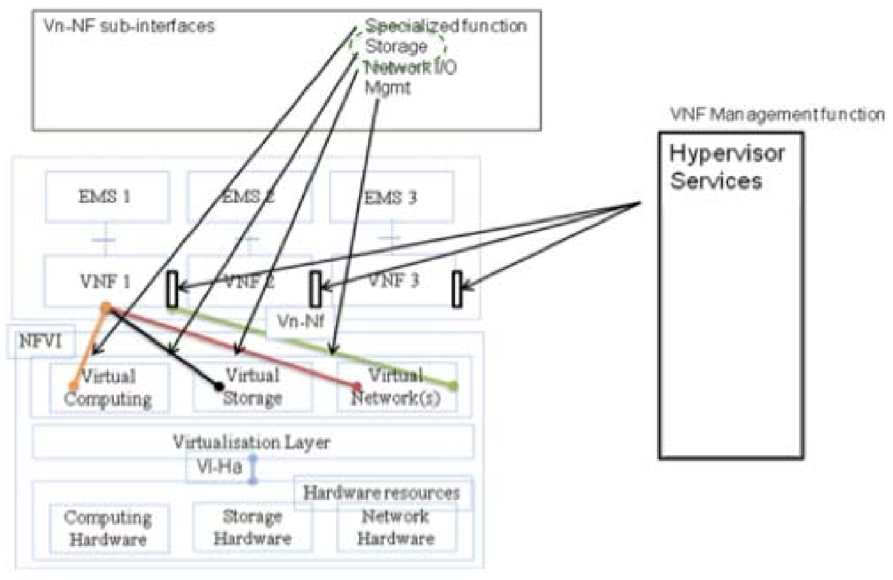

Figure 19: The storage Vn-NF sub-interface is for provisioning the storage that is allocated to the NF or VNF

The storage interface in the figure can encompass a configuration API, as well as any API that may be a part of the protocol itself, such as is the case with object storage.

7.3.5 Security

Due to the shared nature of storage within a virtualisation environment, it should be understood that security is a layered structured in that there is not one set type of security to be implemented across physical and virtual domains. Similarly, different types of services will require different types of storage, which will also bring their own unique security aspects. The scope and combinations of security considerations is too large for any one document, so this will touch on just the key considerations.

Within a virtualisation environment, one disruption or destruction of data can result in bringing down all storage for all virtual machines. This can result from unauthorized accesses to the infrastructure, from authorized individuals performing unauthorized actions, as well as from authorized personnel performing unintended actions as a result of manual manipulations. All of these scenarios shall be accounted for and recovery point capabilities shall be designed in, as well as ways to detect and to account for threats.

7.3.5.1 Auditing Framework

The underlying physical storage shall employ some type of auditing framework. This auditing mechanism will consist of the tracking of root or user level accesses to the NFVI's file system for the deletion or modifications of data via user manipulations. The records of such actions shall be maintained within protected binary logs so that the record of malicious actions cannot be deleted from the logs.

7.3.5.2 Protecting Data at Rest

It should be understood that breaches will occur and that it is only a matter of time before there is an occurence. To protect against malicious destruction of data or access to sensitive data, their needs to be a mechanism in place that can react to the detection of a breach and to isolate the breach from doing damage to persistent data. One such way of accomplishing this is to utilize encryption of the associated volumes with something such as Linux Unified Key Setup (LUKS).

LUKS works by encrypting the superblock of a mounted volume. When the key exchange is successful, the superblock is recognizable and the volume can be mounted and accessed by the system. LUKS can be used in conjunction with physical volumes, such as those managed by Logical Volume Manager (LVM) or physical partitions; and LUKS can also be used in conjunction with mounting file-based volumes, such as virtual disk images.

When a breach is detected anywhere in the system, it should trigger a response that results in the unmounting of attached, or potentially exposed volumes so that data can neither be destroyed, altered or stolen. Once the perceived threat has been neutralized or deemed safe, all unmounted volumes can then be mounted again.

Protecting data at rest via encrypted volumes does not result in any performance overhead as inflight data does not experience a level of encryption, apart from any encryption methods that are otherwise imposed. However, during steady state operations, mounted volumes are effectively unencrypted so that they can be read and written to by system level software and even by users. It is only when a volume is unmounted that its encryption is actually effective.

Additionally, certain otherwise routine commands should be altered or removed eliminate unauthorized deletion or modification to data. For example, commands such as "rm" (delete) and "dd" (used for writing over blocks) and "fdisk" (used for altering disk partitions) should require at least some additional level of approval during steady state deployment. Commands such as "cat", "tail", "more" and "vi" can be used for reading data files and hence can be used for accessing sensitive data at a command line via a user shell.

The most complicated part of protecting data at rest is where and how to provide the key pairs utilized for mounting the protected volumes. It is expected that some level of automation will be desireable, otherwise the daunting task of mounting thousands of volumes will be overwhelming. Utilizing a local device such as a USB drive will not protect data against physical theft. Also, loss of the key device could jeopardize all data. Utilizing a key service can accomplish the purpose of allowing volumes to be remounted in the event that the environment has been deemed safe.

7.3.5.3 Protecting Data In Flight

Protection of in-flight data can be characterized and protected through a number of different means, including:

1) Encrypted networks: Tunnels such as via SSH (Secure Shell) or IPSec (Internet Protocol Security).
2) Web-based applications: SSL (Secure Socket Layers).
3) Package contents: PGP (Pretty Good Privacy).

7.3.5.4 LUN Mapping and Access Control Lists

Physical volumes that are shared throughout the NFVI will typically come as a result of some type of SAN implementation. While the protocol may vary for each implementation, most SAN solutions will invoke or at least make provision for some type of physical mapping of a specific volume to one or more specific client machines, which can also be virtual in nature. FC and SAS adapters (aka initiators) implement unique serial numbers or worldwide names, similar to an Ethernet's MAC address. iSCSI initiators, on the other hand, utilize a reassignable name known as an IQN (iSCSI Qualified Name). Physical LUNs are assigned to specific clients via these unique or pseudo-unique names. Unless these mappings are assigned to the underlying LUN, the volumes can be undiscoverable at most, or in accessible at the least.

While these conventions are for block-based storage, similar conventions can be implemented for file-based storage through the use of mechanisms such as /etc./hosts-allow, /etc./hosts-deny, iptables and other mechanism. The basic premise is that lists can be created to allow certain accesses. Any access not on the list will be denied. And for finer grain control, the hosts-deny list can allow access by all hosts on a particular domain with the exception of those contained on its list.

7.3.5.5 Authentication

In addition to Access Control Lists, authentication can also be utilized. Authentication can come in the form of a name and pass phrase.

7.3.5.6 Physical Storage Nodes Within the NFVI Node

Storage may be physically implemented in a variety of ways. It could, for example be implemented as a component within a Compute Node. An alternative approach may be to implement Storage Nodes independent of the Compute Nodes as physical nodes within the NFVI Node. An example of such a Storage Node may be a physical device accessible via a remote storage technology as discussed in clause 7.3.3.1.2 (Remote Storage).

The NFVI is required to continue to provide end-end service despite the existence of failures within physical Storage Nodes and the connectivity to those Storage nodes. (See ETSI GS NFV-REL 001 [i.3]). In practice, this requires the Storage Nodes to have multiple connections to independent (spatially diverse) network nodes within the NFVI Node. The figure below illustrates this aspect of Storage Node connectivity within the NFVI Node.

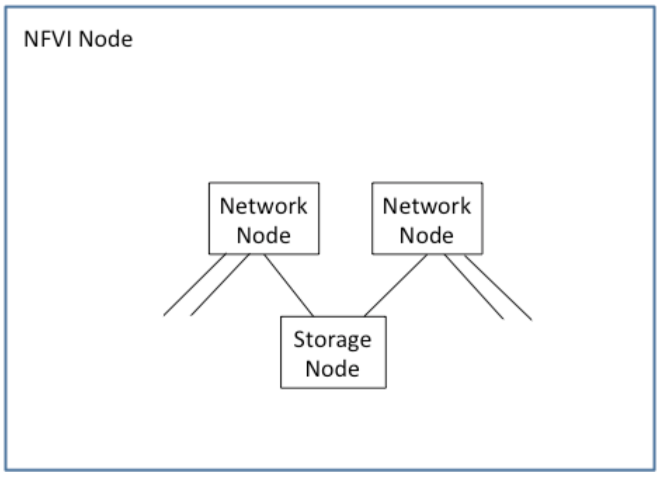

Figure 20: Storage Nodes shall have more than one network connection within the NFVI Node

7.4 Hardware Resource Metrics of the Compute Domain

An objective of the compute domain is to provide a library which exposes hardware characteristics of the compute node in "real-time". The VIM communicates directly to the compute domain and through the hypervisor to access all the hardware metrics. Metrics can be static or dynamic in nature.

Clause 5 describes requirements for exposing hardware characteristics (i.e. metrics) for planning/provisioning and real-time monitoring/deployment of VNFs. Interfaces to pass metrics to the VIM are described in the NFVI domains and interfaces (NFV Infrastructure Architecture).

To be useful by MANO metrics shall be processed in SWA using an information model/s to produce KPIs (Key Performance Indicators).

7.4.1 Static Metrics for FNF Capacity and Capability Characteristics

Static metrics expose compute node characteristics which do not change or change slowly (e.g. once a day). These metrics ultimately act as a first order filter for selecting/provisioning a node for deploying a VNF. Static metrics are obtained from reading various OS and ACPI tables. The Advanced Configuration and Power Interface (ACPI) specification provides an open standard for device configuration and power management by the operating system). Additional metrics may be stored in local structures provisioned by the vendor or administrator. For example compute node performance index, energy efficiency index, geographic location, specific features enabled/supported, security level, etc.

Table 6: Static hardware metrics

Resource	Metric/s	Example/s and units
Hardware platform	Asset Tag Device ID GUID Manufacturer Firmware/Software information Management Controller ID	Information required to remote configure and administer the machine Rack-mount-server/HP/ProLiant DL380p/Gen8
	Sockets	4
	System power profile (usage-model dependent)	Total system throughput/power (at max throughput). This is an empirical number set by administrator of the node
CPU	Processor family (includes vendor), model, version, frequency, number of physical and logical cores	CPUID
	Instruction Set Architecture information	x86, ARM
	Special features e.g. cQoS	
	Power states supported	C-states, P-states
	Physical socket location	Socket 1
	Cache size, line size, cache level and type, read and write policy	
	Health status	
Chipset	Direct IO access to processor cache	DDIO
	I/O virtualisation	VT-d
	DMA type	CBDMA (Common buffer DMA)
	Security (e.g. trusted boot)	TXT, TPM Version, Class of security (low, moderate, high)
	RAS features	
Memory subsystem	Type	ECC, non-ECC, DDR3, etc.
	Amount of physical memory present on the host	Gigabytes (GB)
	Performance	IOPS per TB
	Amount of memory available to run virtual machines and allocated to the hypervisor	
	Amount of memory allocated to the service console	
	Channels	
	Temperature	
	Power consumption and power saving modes	
	Interleaving configuration	
	Memory RAS features	
	Memory collection health status (memory module has failed or is predicted to fail, memory board error, memory redundancy degraded, memory recovered from degraded redundancy)	
Network I/O	Aggregate bandwidth	100 Gbps
	PCIe slots populated	
	Network Adapter: • Type and model • Flow affinity/steering e.g. SR-IOV • NIC acceleration e.g. TSO • RAS features	
	Utilization	
	Max Power Consumption	
	Temperature	
Accelerators	Crypto	Number of crypto accelerators, Vendor, model, number of units, max throughput for one algorithm
	Compression	Number of compression accelerators, Vendor, model, number of units, max throughput for one algorithm
Local storage	Type/capacity	Storage media type, Hard disk speed

7.4.2 Dynamic Metrics for VNF

An orchestrator can identify a candidate platform based on static metrics however to actually instantiate an VNF additional dynamic metrics will be required, e.g. CPU, Memory, IO headroom currently available. These metrics could be provided on a per-query basis or the compute node could proactively update hypervisor and software domains at regular intervals.

Table 7: Dynamic hardware metrics

Resource	Metric/s
CPU	Currently available cores
	Idle and operating power state residency, e.g. C1+ state residency, P state residency
	Per core temperature
	Cache utilization, LLC misses, TLB misses
	Time to Failure
	Error-free processing time
Memory subsystem	Bandwidth utilization
	Balance
	Thermal throttling
Virtualisation	VMExits, I/O page misses, e.g. VT-d page misses
RAS	Number of faults/sec
I/O	I/O b/w utilization
	Number of interrupts to hosts and guests

As Activation/Creation/Setup is an important aspect in continuing operations, the result are the following metrics (an initial proposal) to quantify this function.

Table 8: Dynamic hardware Activation metrics

Resource	Metrics	Examples and Units
Physical Server Blade	Initialization Time	Time from power activation until "in-service", informing all necessary managers and orchestrators that the resources on the server/blade are ready for commissioning (*in the form of an operating Hypervisor ready to instantiate a VM to serve a VNFi*.
	Failed Initializations	Count of attempted Initializations that do not result in the server/blade reaching the state of readiness for commissioning.
	Successful De-activation Time	Time from "in-service", until notification of de-activation informing all necessary managers and orchestrators that the resources on the server/blade are de-activated and should be removed from resource pools.
Storage Unit (i.e. disk)	Activation Time	Time to fully stable, active use in array (*such as adding/replacing a member of a RAID array*).
	Failed Activations	Count of attempted Activations that do not result in the storage unit reaching the state of readiness.
	Successful De-activation Time	Time from "in-service", until notification of de-activation informing all necessary managers and orchestrators that the resources on the storage unit are de-activated and should be removed from resource pools.

If examining the 3x3 matrix for CPU-related dynamic metrics, the result is the following coverage of functions and criteria.

Table 9: Matrix of Dynamic CPU-related Physical Resource Metrics

	Speed	**Accuracy**	**Reliability**
Activation	Initialization time.		Failed Initializations,
Operation	Available core count Per core: temperature Idle power state residency Operating voltage/frequency point residency Cache utilization,	LLC misses, TLB misses,	Time to Failure Error-free Processing Time
De-activation/Deletion/Take-down	Successful De-activation time (Blade or Storage Unit)		

8 Interfaces within the Domain

8.1 PCIe

The PCIe bus interconnects the NIC and/or acceleration cards to the host CPU. It needs to support a large bandwidth, usually equivalent to the network interfaces bandwidth of the NIC.

8.2 SR-IOV

The SR-IOV protocol is used to virtualize the PCIe and the attached NIC. The result is that one physical NIC can support up to 128 virtual functions (VFs).

Single Root I/O Virtualisation (SR-IOV) defines a method to split a device into multiple PCI Express Requester IDs in a fashion that allows an I/O MMU to distinguish different traffic streams and apply memory and interrupt translations to these streams. The result is that these traffic streams can be delivered directly to the appropriate Virtual Machine (VM), and in a way that prevents non-privileged traffic flows from impacting other VMs. Each PCI Express virtual function can be directly assigned to a Virtual Machine (VM), thereby achieving near native performance. SR-IOV enables network traffic to bypass the software switch layer and the virtual function to be assigned to the VM directly. By doing so, the I/O overhead in the software emulation layer is diminished.

In Pass Through/SR-IOV based virtualisation, the management OS can configure the Virtual Functions (VFs) and assign them to particular VMs. The VFs are exposed as hardware devices to the VM. All data packets flow directly between the guest OS and the VF. This eliminates the software path between the Management OS and the VM for data traffic. It bypasses the Management OS's involvement in data movement by providing independent memory space, interrupts and DMA streams for each VM. Frames are sent to the external network via the physical port of the device or to another VM via the internal port connected to the VF. In all cases, it eliminates the need for any involvement of the Management OS in the data path resulting in: improved I/O throughput and reduced CPU utilization, lower latency, improved scalability.

8.3 RDMA and RoCE Support

This clause discusses Remote DMA (RDMA) support, including RDMA over Infiniband and RoCE (Remote DMA over converged Ethernet).

Remote Direct Memory Access (RDMA) is the remote memory management capability that allows server-to-server data movement directly between applications Memory without any CPU involvement. RDMA has been shown to deliver value propositions that are not available through any other communications standards, including low latency, improved resource utilization, flexible resource allocation, scalability and unified fabric.

RDMA over Converged Ethernet (RoCE) is a standard that defines a new RDMA protocol over Ethernet. With advances in data center convergence over reliable Ethernet and Data Center Bridging (DCB), RoCE uses the efficient RDMA mechanism to provide lower CPU overhead and increase mainstream data center application performance at 10GigE and 40GigE link speeds and beyond. Running RDMA applications in an Ethernet infrastructure enable application performance, efficiency, and cost and power savings that come from the reduction in application latency. RoCE communications can be as low as 1/10th the latency of that of other standards-based solutions. Adopters of RoCE can make use of RDMA's capabilities without leaving the familiar transport and network management system of Ethernet. In this way, adopters can upgrade their application performance without investing in alternative switching interconnect.

8.4 InfiniBand

The InfiniBand Architecture emerged in 1999 as the joining of two competing proposals known as Next Generation I/O and Future I/O. These proposals, and the InfiniBand Architecture that resulted from their merger, are all rooted in the Virtual Interface Architecture (VIA). The Virtual Interface Architecture is based on two synergistic concepts: direct access to a network interface (e.g. a NIC) straight from application space, and an ability for applications to exchange data directly between their respective virtual buffers across a network, all without involving the operating system directly in the address translation and networking processes needed to do so. This is the notion of "Channel I/O" - the creation of a "virtual channel" directly connecting two applications that exist in entirely separate address spaces.

InfiniBand is often compared, and not improperly, to a traditional network such as TCP/IP/Ethernet. InfiniBand provides a messaging service that applications can access directly. The messaging service can be used for storage, for InterProcess Communication (IPC) or for a host of other purposes, anything that requires an application to communicate with others in its environment. The key benefits that InfiniBand delivers accrue from the way that the InfiniBand messaging service is presented to the application, and the underlying technology used to transport and delivers those messages. This is much different from TCP/IP/Ethernet, which is a byte-stream oriented transport for conducting bytes of information between sockets applications. Furthermore, InfiniBand Network Interconnect delivers higher bandwidth and lower latency compared to available TCP/IP/Ethernet network Interconnect.

8.5 DPDK & ODP Support

In x86-based designs there is a need to support the Data Plane Development Kit (DPDK) for the physical NIC, physical accelerator, and their virtual functions.

In ARM-based designs, the Linaro initiative's Open Data Plane (ODP) is to create an open standard for accessing data plane services in a way that is compatible with DPDK and the ARM partner solutions.

9 Modularity and Scalability

9.1 Modularity and Scalability within the Domain

The compute domain should be architected such that a distributed virtual appliance can be hosted across multiple compute platforms, as shown in figure 14.

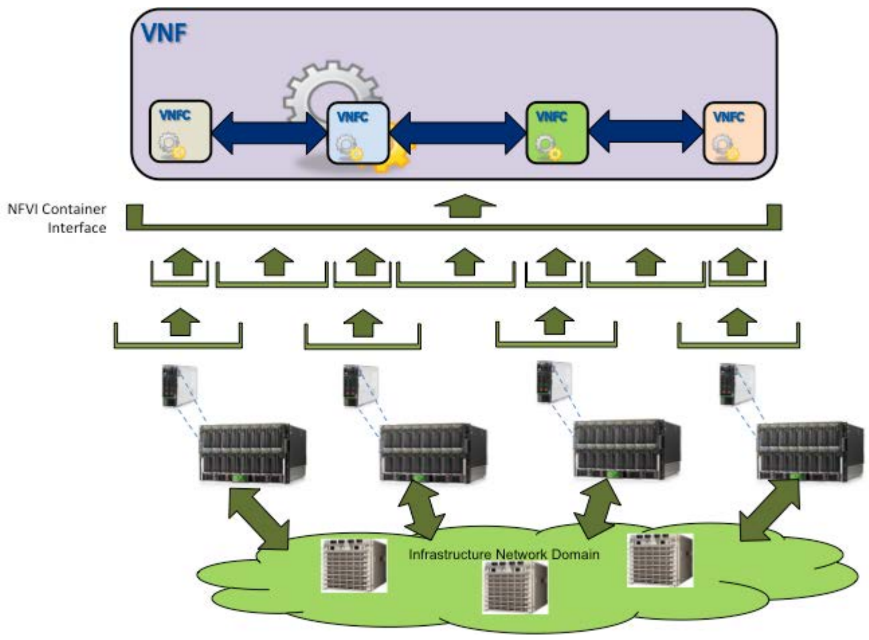

Figure 21: Features of the Domain supporting end-to-end scalability

The Compute Domain may be deployed as a number of physical nodes (e.g. Compute Nodes and Storage Nodes) interconnected by Network Nodes within an NFVI Node. Figure 15 gives an illustrative example of how an NFVI node might be constructed from Compute, Storage, Network and Gateway nodes. In this example, a Compute Node might be implemented as a server blade, a Network Node might be implemented as a top of rack Ethernet switch, a Storage Node might be implemented as a network attached storage device and a Gateway Node might be implemented by some optical interface device.

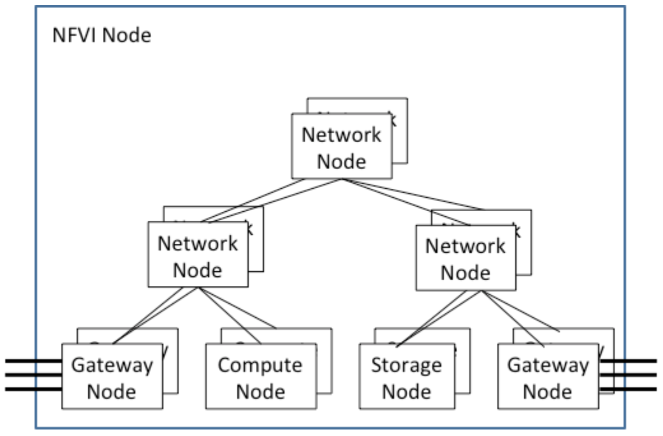

Figure 22: Example NFVI Node implementation including Compute Nodes and Storage Nodes

9.2 NFVI Scale Implications for Compute Domain

The NFVI is deployed as NFVI Nodes at locations called NFVI-PoPs which would be expected to contain Compute Domain functions. The number and location of NFVI-PoPs depends on the deployment. A variety of different deployment strategies are possible (see e.g. [i.1]). One approach is to extrapolate from existing Network Element deployments (in N-PoPs) and assume a 1:1 mapping (See ETSI GS NFV 001 [1] for discussion of 1:1 and alternative mappings between NE'S to VNF'S) between N-PoPs and NFVI-PoPs. The table below shows the potential scale of the NFVI, and consequently the potential distribution of Compute Domain functions. Table 10 provides illustrative examples, but the numbers of different types of NEs and location naming schemes etc. may vary by network operator.

Table 10: Potential NFVI scale

# N-PoPs	Example Location Types	Example NE types	NFV Use cases
1 - 10	IT Data Center	firewall	NFVIaaS, VNFaaS, VNPaaS, VNF Forwarding Graph
10 - 100	Major Central Office	router	vIMS
100 - 1 000	Minor Central Office	BRAS	
1 000 - 1 000 000	Curbside Cabinet, Cell Tower	eNodeB, DSLAM	vCDN vMobile Base Station
1 000 000 +	Subscriber Premises	CPE, Mobile Devices, IoT	vCPE, vFixed Access Network

9.3 Modularity of NFVI Node Hardware Resources

NFVI Node hardware resources include compute, storage and network functions. The relative capacity of these functions to be implemented in particular deployment units is a subject for further study. The dashed lines in figure 23 indicates that these are functions contained within the NFVI hardware resources, and not field replaceable units.

The modularity of the NFVI hardware resources is concerned with the identification of the replaceable units. The figure below illustrates the basic modularity options in terms of replaceable units: the NFVI Node can be composed of COTS field replaceable units; the NFVI Node can have no field replaceable units, or it can be a fields replaceable unit for some other entity.

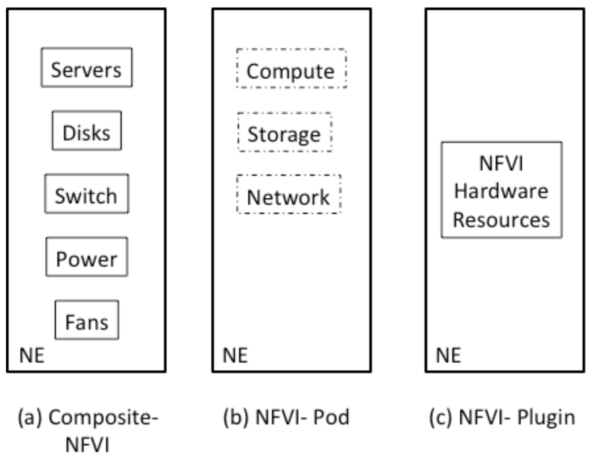

Figure 23: NFVI hardware resources modularity options

Composite-NFVI (illustrated in figure 23(a)) would compose the NFVI from COTS elements such as servers, disks, switches, power supplies, fans, hardware acceleration modules, line interface variants, etc. In this case the field replaceable units are components of the NFVI. Another implementation approach might be the use of an OCP pooled resource structure. Operators typically maintain an inventory of field replaceable units in order to effect repairs in the event of failures. The use of COTS components to construct the NFVI (compared to proprietary hardware) reduces the variety of field replaceable units that shall be maintained and tracked in inventory.

An **NFVI-Pod** (illustrated in figure 23(b)) is a COTS deployable unit of NFVI hardware resources with no field replaceable units. NFVI-Pods could be deployed with various capacities and form factor, from small e.g. pole mounted devices to shipping containers. Identification and specification of capacities and form factors requires further study beyond the scope of this specification. An objective of NFV is to convert failure events into capacity reductions. The NFVI-Pod extends this approach to eliminating the need for inventorying field replaceable units that are components of the NFVI, and only inventorying deployable NFVI-Pods. Failed elements within an NFVI-Pod would be administratively marked as unavailable for use.

An **NFVI-Plugin** (illustrated in figure 23(c)) is a COTS deployable unit of NFVI hardware resources that is a field replaceable unit within some other network element. The NFVI hardware resources of the NFVI-Plugin are managed by the NFV Management and orchestration entities which may be independent of the EMS and NMS that may be managing the NE hosting the NFVI-plugin.

NFVI Components are NFVI hardware resources that are not field replaceable, but are distinguishable as components at manufacturing time. An integration of NFVI hardware resources in a System on a Chip might be an example of this approach to modularity. Like an NFVI-Plugin, an NFV Component is a part of a larger physical device. Further study of NFVI components is beyond the scope of this specification.

10 Features of the Domain Affecting Management and Orchestration

The features of the storage domain affecting management and orchestration are described in clause 5.3.1 (Managing NFVI elements).

10.1 Arbitration of hardware accelerators

A number of the opportunities for vendor differentiation, within the NFVI, revolves around the provisioning of hardware, which can provide acceleration of particular functions required by VNFs (e.g. encryption or packet header offload). Often, the most computationally efficient method to employ these accelerated functions is to pass control of the relevant hardware resource into the VM, mapping memory directly, and bypassing any hypervisor control. There is, however, a security trade-off associated with this method in that it reduces the ability of hypervisors to control resources or to ensure that they are neither being misused nor impacting the ability of other VMs (in this case, VNFCIs) to function correctly and reliably. Realizing that not all accelerated NFVI computing infrastructure will involve the use of a hypervisor (e.g. bare metal or Linux Containers), it is important for such bindings to be managed and tracked and for there to be some form of arbitration in the event a hardware resource is in current use by a particular VNF.

10.2 Life Cycle of NFVI Nodes and their Replaceable Units

After deployment of a new NFVI Node, the resources provided by that NFVI Node shall be included into the inventory repository of NFVI hardware and software resources of the VIM administering that NFVI Node. Similarly when a NFVI Node is removed from service, the corresponding resources should be removed from the inventory repository. The entries in the resource catalogue can be provided with a variety of mechanisms. The ongoing synchronization of the inventory repository with actual physical installations is a pre-requisite for efficient orchestration of those resources.

NOTE: Mechanisms for automating the discovery of NFVI Nodes, their attachment to MANO entities and synchronization with respective inventory repositories is for further study.

In the case of a Composite NFVI Node, there are replaceable units within the NFVI Node. A similar synchronization problem occurs between the resource catalogue and the physical compute nodes, storage nodes, etc. that have been deployed within that Composite NFVI Node.

11 Features of the Domain Affecting Performance

11.1 Matching Resource Requirements and NFVI Compute Server Capabilities

ETSI GS NFV-PER 001 [7] provides the list of VNFCI requirements that should be included in the VM Descriptor (VMD) template, and the list of HW capabilities that should be included in the Compute Host Descriptor (CHD) which is used by the VIM to build the inventory repository of NFVI hardware and software resources. By using both VMD and CHD, the VIM can have the needed information to assign the right resources in order to assure predictable high performance. NFV PER GS document and Architecture of Compute Domain clause 8 - "Templates for portability" contain more details regarding the CHD and VMD.

11.2 Performance Related Workload Categories

ETSI GS NFV-PER 001 [7] defines the following Performance related workload categories (related to Compute, Network, Storage domains and the addition of CRAN DSP processing).

- **Data plane workloads**, which cover all tasks related to packet handling in an end-to-end communication between edge applications. These tasks are expected to be very intensive in I/O operations and memory R/W operations.
- **Control plane workloads**, which cover any other Network Functionality that is not directly related to the end-to-end data communication between edge applications. These tasks are expected to be very intensive in CPU processing and memory R/W operations.
- **Storage workloads**, which cover all tasks related to disk storage and are very I/O intensive.
- **Signal processing workloads**, which cover all tasks related to digital processing such as the FFT decoding and encoding in a C-RAN Base Band Unit (BBU).

The present document defines how the above workloads are supported by the NFVI. It further defines HW Acceleration classes supporting the above workload classifications. NFVI HW Acceleration Abstraction defines the mechanisms by which these workload processing capabilities are uniformly provided, abstracting their actual implementation.

11.3 Other performance related concepts

11.3.1 Compute Domain Performance Related Features

There are a number of features from HW, Virtualisation SW, Acceleration technology & SW design which affect performance. The compute node capabilities need to be reported to the VIM via the CHD. Effective handling and allocation of Compute Domain capabilities, and matching VM requirements greatly affect performance.

- HW Implementation
  - CPU Architecture and basic performance levels - PCI-E, Infiniband, clock speed, Number of Cores, etc.
  - NFVI Compute Node architecture and connectivity - Bus structure and BW, NIC performance, Memory and Storage structure.
  - Cache structure and sizes.
  - Support for large memory pages and extended TLB caches with large pages.
  - Support for SR-IOV (classification of packets in independent, per VM TX/RX queues) and other traffic & data optimization technologies.
  - Direct I/O access to processor memory & OS, e.g. Architectures that support DMA (Direct Memory Access) and RDMA.
  - IOMMU or translation services for I/O.
- Acceleration Technologies
  - On Chip HW based acceleration - e.g. AES, CRC, Cryptography, Transcoding.
  - Compute Intensive acceleration - e.g. Heterogeneous Computing/GPU.
  - Compute Acceleration Pool.
  - Network Intensive function acceleration - e.g. NP, FPGA, CPU based support for data plane workload acceleration and data traffic optimization - e.g. NAT, ACL, DPI.
  - Storage Acceleration - e.g. Storage Clusters.
  - NIC based acceleration - e.g. SR-IOV, vSwitch Bypass, Network Intensive processing.

11.4 QoS metrics

QoS metrics are the core network performance metrics based on network measurements to control the SLA offered by the NFVI Compute domain and QoE level of applications and services. Examples include:

- Throughput.
- Latency.
- Frame Loss Rate.
- Back-to-Back Frame Rate.
- Packet delay variation.
- Service Disruption Time for Fail-over Convergence.

11.5 Performance Monitoring

Performance monitoring are a set of methods and technologies to Monitor and report on NFVI performance, to be used in capacity planning, dynamic Performance monitoring, SLA/QoE guarantee, and differentiated services and monetization. Examples include:

- Run time Performance Testing & monitoring.
- Failover Convergence time.
- VM Migration Performance.
- DP performance - On going testing.

11.5.1 Conclusion - Features of the Domain Affecting Performance

VNFCIs require a set of capabilities from the NFVI, some of those relate to the NFVI compute domain performance levels and predefined QoS metrics. These VNFCI requirements translate relate to Data, Compute, Storage and Signal processing workloads and are reported to the NFV Management & Orchestration via the VNF Descriptor.

NFVI VMs support VNFCI requirements by providing virtualisation of compute domain resources, OS and SW APIs, including acceleration technologies. In turn, the compute domain supports the VM requirements by providing compute and storage in the form of HW Implementation, including acceleration technologies. These compute domain capabilities are reported to the NFV Management & Orchestration via the CHD. NFVI compute nodes ability to maintain their announced performance levels is verified by a set of performance monitoring capabilities supplied by the NFVI.

12 Features of the Domain Affecting Reliability

The Compute Storage Domain supplies the fundamental computation and related capabilities to other Infrastructure domains and to all virtual resources, thus it has a direct effect on their reliability.

The domain will consist of multiple Compute Nodes (coherent execution environments with memory, NICs, and possibly other elements) which have various degrees of dependency on physical/common components of the domain: chassis, power supply, cooling, etc. There would be a multiplicity of Storage Nodes (and Acceleration Nodes when present) with similar dependence on common components. Thus, one of the critical features of the domain is to accurately represent each Compute Node and its associated common hardware in the management inventory (which resides in the Virtual Infrastructure Manager). Another feature is to communicate this information promptly when new Nodes are powered-up and ready for service, or when the Nodes fail or are removed from service (powered-down), thereby keeping the inventory synchronized with truth for resources at the VIM and any Load-balancing functions.

The associated Virtual Infrastructure Manager (VIM) is responsible for maintaining the inventory of each domain. Thus, the VIM should not reside on common components within the Nodes it is managing (as discussed in clause 5.2.1).

A related feature of the domain support for VIM inventory is its ability to represent affinity among compute nodes (multiple components of the compute & storage domain) with respect to their dependencies on other elements of the domain, so that the diversity and placement policies of requests for resources can be satisfied accurately:

- Rack.
- Fabric.
- TOR switch.
- Power distribution.
- Rack shelf.
- Server chassis.
- Storage chassis.

VNFs should be distributed among spatially diverse clusters or pools of nodes, such that failures result in temporary capacity reduction, not unavailability.

Within the Compute, Storage, and Accelerator nodes, various features are can be activated to improve Reliability, Availability, and Serviceability, or RAS. The following table summarizes the features by node type. See ETSI GS NFV-REL 001 [i.3] clause 12.4.1 for additional explanation of many of the features below.

Table 11: Features by Node type

Compute	Spare processors and live migration
	Self-healing bus
	Memory mirroring
	Spare memory and migration
	Processor and memory hot add
	Corrected and Uncorrected error monitoring and logging (memory and cache error checking and correction)
	Failure anticipation/prediction and mitigation using features above (clause 9.3 of ETSI GS NFV-REL 001 [i.3])
	Additional: parallel computation and checking, corrupted data containment, HW partitioning.
Storage	Reliable duplicated storage, Local and Remote, networking for LAN and WAN
	Media: Hard Disk Drive, Solid State Disk, Hybrid of HDD and SSD
Common	Uninterruptable Power Supply, spare cooling resources

Network services often demand lesser or greater levels of Availability, which have been categorized in ETSI GS NFV-REL 001 [i.3]. The three levels described there can be achieved using various combinations of the features above.

Heartbeats and time-outs are critical tools to determine liveliness on failures. Clauses 10.4 and 10.5 of ETSI GS NFV-REL 001 [i.3] describe the requirements for these liveliness features and Hypervisor or VIM monitoring.

Each Node in the Domain should have multiple connections to Network Nodes, and the connections should be spatially diverse to avoid single points of failure.

It may be possible to distinguish management connectivity failure from Node failure in the VIM, where information on the liveliness of and utilization of associated resources is present. For example, it may be possible to detect that the Virtual Network (tunnel or VLAN) connectivity has failed in addition to expiration of the watch-dog timer for a specific node.

In summary, higher reliability can be achieved through both diversity and accurate inventory management, in addition to methods to improve RAS for individual resources.

13 Features of the Domain Affecting Security

Please refer to clause 7.3.6 on storage security.

13.1 Physical security

The NFVI provides the underpinnings for the services in turn provided by Virtual Network Functions, and shall, as such, be protected from attack. Physical security is therefore very important, as attacks on the physical infrastructure may have impact on the reliability, integrity, performance, confidentiality and availability of those services. Since the NFVI spans many locations, physical security is difficult, but important to manage in order to protect the integrity of the domain. Making use of tamper-evidence technology on chassis or BIOS notifications, for hardware which is now more user-serviceable than previously shall be balanced against the impact on maintenance and upgrade activities.Standard techniques are available for use in data centres but more remote locations present new challenges, particularly when unstaffed, such as towers, exchanges or commercial customers. Whereas previously, the hardware providing network functions was dedicated and not easily re-purposable, the replacement of this hardware with COTS hardware, generally of high performance specifications, is likely to present a tempting set of targets for thieves. Equally, attackers are likely to find it easier to compromise such hardware than previously. Remote monitoring and tamper-evidence is important in this context.

13.2 Security devices

There are a number of hardware security devices which reside within the Compute domain, and which may be used either by VNFs, or by various other parts of an NFV deployment. While not required infrastructural components, the following may be deployed:

- HSM (Hardware Security Module) - provides cryptographic operations and protected key storage.
- TPM (Trusted Platform Module) - provides certain cryptographic capabilities useful for secured boot and related activities.
- Cryptographic accelerator - provides offload of certain cryptographic operations to dedicated hardware.
- Protected execution environment - technology to allow VMs to execute some part of themselves within a portion of memory protected from introspection by the hypervisor.

13.3 Hardware or Device-based Authentication

Just as the use of a catalogue may be essential for storing trusted images representing VNFs, there may be need for a registration database to track trusted hardware components that may be used to establish multi-level authentication. In such a deployment model, a token may be validated against a particular hardware signature such as a MAC address or serial number. Used in conjunction with an approved image and an approved interface, the entirety of the communications from the VNFC to the supporting NFVI can be established. In the event that a physical theft of a hardware component occurs, it can be removed from the database of trusted components so that any attempted usage can be blocked.

While there are a number of ways to achieve device/hardware based authentication, it is imperative to deploy such technology in conjunction with some form of registration and token management to avoid orphaned keys from presenting potential security vulnerabilities.

13.4 Tunnelling isolation in hypervisor and non-hypervisor container environments

In the event that tunnels or VPNs are utilized by VNFIs within VMs (or their equivalent in Linux Container and bare metal deployments), the use of drivers in the "host" kernel drivers should be avoided in order to ensure isolation between the user and kernel space and to prevent the application from causing a segmentation fault or kernel panic of the underlying host operating system. In environments where a hypervisor is used, it is assumed that the kernel driver for the tunnel will run within the virtual machine, and thus avoid causing instability of the underlying host.

Annex A (informative): Authors & contributors

The following people have contributed to this specification:

Rapporteur:

- Nabil G. Damouny, Netronome

Other contributors:

- Michael Young, Huawei (Associate Editor)
- Zvika Bronstein, Huawei
- Trevor Cooper, Intel
- Rob Dimond, ARM
- Jan Ignatius, NSN
- Ramki Krishnan, Brocade
- Kin-Yip Liu, Cavium
- Alfred Morton, AT&T
- Rich Phelan, ARM
- Andy Reid, BT
- Mattias Rimbark, Ericsson
- Jim Scott, EZchip
- Stephen Shew, Ciena
- Steven Wright, AT&T

Annex B (informative): Bibliography

- NFV White paper: "Network Function Virtualisation", issue 1 (2012).
- USAGE: Standards Units of Measure for IaaS, Rev 1.1 2013, Open Data Cneter Alliance.

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

- 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.

History

Document history		
V1.1.1	December 2014	Publication