ETSI GS NFV-INF 004 V1.1.1 (2015-01)

Network Functions Virtualisation (NFV);
Infrastructure;
Hypervisor Domain

Disclaimer

Reference
DGS/NFV-INF004

Keywords
interface, NFV

ETSI

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

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

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

Contents

Intellectual Property Rights

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

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

Foreword

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

Infrastructure Architecture Documents	Document #
Overview	GS NFV INF 001
Architecture of Compute Domain	GS NFV INF 003
Architecture of Hypervisor Domain	GS NFV INF 004
Architecture of Infrastructure Network Domain	GS NFV INF 005
Scalability	GS NFV INF 006
Interfaces and Abstraction	GS NFV INF 007
Test Access	GS NFV INF 009

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 the architecture of the Hypervisor Domain of the NFV Infrastructure which supports deployment and execution of virtual appliances. The present document will primarily focus on the use of hypervisor for virtualisation, due to time and resource constraints, However, the hypervisor requirements are similar if not the same for implementing linux containers or other methods for virtualisation.

NOTE: From WikiArch: "Linux Containers (LXC) are an operating system-level virtualisation method for running multiple isolated server installs (containers) on a single control host. LXC does not provide a virtual machine, but rather provides a virtual environment that has its own process and network space. It is similar to a chroot, but offers much more isolation".

There needs to be further research w.r.t to Linux Containers, including developing the ecosystem.

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

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

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-INF 001 (V1.1.1): "Network Functions Virtualisation (NFV); Infrastructure Overview".

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] ETSI GS NFV 004: "Network Functions Virtualisation (NFV); Virtualisation Requirements".

[i.2] IETF RFC 4133: "Entity MIB (Version 3)".

[i.3] IEEE 802.1D™: "IEEE Standard for Local and Metropolitan Area Networks -- Media access control (MAC) Bridges".

[i.4] IEEE 802.1Q™ MIB: "IEEE Standard for Local and Metropolitan Area Networks, Management Information Base".

[i.5] IETF draft-ietf-opsawg-vmm-mib-00: "Management Information Base for Virtual Machines Controlled by a Hypervisor".

NOTE: Available at http://tools.ietf.org/html/draft-ietf-opsawg-vmm-mib-00.

[i.6] IEEE 1588™: "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems".

[i.7] IEEE 802.11™: "Wireless LANS IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN".

[i.8] IEEE 802.3ad™: "Link Aggregation".

[i.9] IEEE 802.3™ MIB: "Link Aggregation, Management Information Base".

[i.10] Hotlink: http://www.virtualizationpractice.com/hotlink-supervisor-vcenter-for-hyper-v-kvm-and-xenserver-15369/.

[i.11] Systems Management Architecture for Server Hardware (SMASH).

NOTE: Available at http://www.dmtf.org/standards/smash.

[i.12] NFVINF(13)VM_019_Data_plane_performance.

[i.13] http://pubs.vmware.com/vsphere-51/index.jsp?topic=%2Fcom.vmware.vsphere.networking.doc%2FGUID-E8E8D7B2-FE67-4B4F-921F-C3D6D7223869.html

[i.14] http://msdn.microsoft.com/en-gb/library/windows/hardware/hh440249(v=vs.85).aspx

[i.15] http://www.vcritical.com/2013/01/sr-iov-and-vmware-vmotion/

[i.16] ETSI GS NFV-INF 010: "Network Functions Virtualisation (NFV); Service Quality Metrics".

3 Definitions and abbreviations

3.1 Definitions

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

application VMs: VM not utilizing an OS

hypervisor: virtualisation environment running on a host

NOTE: The virtualisation environment includes the tools, BIOS, firmware, Operating Systems (OS) and drivers.

portability: See ETSI GS NFV-INF 001 [1].

standard: is de-jure, de-facto or open standard that fulfils the requirement

NOTE: This assumption should be applied to all requirements through the entire document.

3.2 Abbreviations

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

API Application Programming Interface
BIOS Basic Input Output System

BSD	Berkeley Software Distribution
CIM	Centralized Interference Mitigation
CLI	Comand Line Interface
CMS	Call Management System
CPU	Compute Processing Unit
DMA	Direct Memory Access
Dpdk	Data plane development kit
EE	Electrical Engineering
FFS	For Further Specification
GUI	Graphical User Interface
HA	High Availability
IETF	Internet Engineering Task Force
INF	Infrastructure
IO	Input Output
IP	Internet Protocol
IPMI	Intelligent Platform Interface
JVM	Java Virtual Machine
KVM	Kernel Virtual Machine
LAN	Local Area Networks
LLC	Lower Level Cache
LLDP	Link Layer Discovery Protocol
LXC	Linux Containers
MAC	Media Access Controller
MANO	Management and Orchastration
MIB	Management Information Base
NF	Network Function
NFVi	Network Function Virtualisation Infrastructure
NFVINF	Network Function Virtualisation Infrastructure
NIC	Network Interface Card
NOC	Network Operator Council
NUMA	Non Uniform Memory Access
OAM	Operations and Maintenance
OS	Operating System
OSS	Operations Systems and Software
OVF	Open Virtual Framework
PAE	Physical Address Extension
PCI	Peripheral Component Interconnect
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Row Address Strobe
RELAV	Reliability and Resiliency Work Group
RFC	Request For Comment
SCSI	Small Computer System Interface
SDO	Standards Development Organizations
SEC	Security Working Group
SLA	Service Level Agreement
SNMP	Signalling Network Management Protocol
SR-IOV	Single Root I/O Virtualisation
SWA	Software Architecture Work group
TCP	Transport Control Protocol
TLB	Translation Lookaside Buffer
UDP	User Datagram Protocol
UUID	Universally Unique Identifier
VIM	Virtualisation Infrastructure Manager
VM	Virtual Machine
VN	Virtual network
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFI	Virtual Network Function Interface
VSCSI	Virtual Small Computer System Interface
vSwitch	virtual Switch
VT	Virtualisation

WG	Working Group
XAPI	eXtended Application Programming Interface

4 Domain Overview

Popek and Goldberg paper 'Formal Requirements for Third Generation Architectures': set the definition of hypervisors in 1974.

- Equivalence: the hypervisor provides an environment for programs which is essentially identical to the original machine.
- Resource control: the hypervisor is in complete control of system resources.
- Efficiency: programs run on this (virtualised) environment show at worst only minor decreases in speed.

Equivalence

The environment provided by a hypervisor is functionally equivalent to the original machine environment. This implies that the same operating systems, tools and application software can be used in the virtual environment. This does not preclude para-virtualisation and other optimization techniques which may require operating systems, tools and application changes.

Resource Control

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

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

Efficiency

Even when not emulating a complete hardware architecture, there can still be aspects of emulation which cause a significant performance hit. Typically, computer architectures provide means to offload these aspects to hardware, as so called virtualisation extensions, the set of operations that are offloaded and how they are offloaded varies between different hardware architectures and hypervisors as innovation improves virtualisation performance.

EXAMPLE: Intel VT and ARM virtualisation extensions minimise the performance impact of virtualisation by offloading to hardware certain frequently performed operations.

There can be many virtual machines running on the same host machine. The VMs on the same host may want to communicate between each other and there will be a need to switch between the VMs.

Infrastructure Domains

Figure 1 illustrates the four domains of the NFV architecture, their relationship with each other and their relationship to other domains outside the infrastructure. The figure also sets out the primary interfaces. Hypervisor for the present document entails tools, kernel, host.

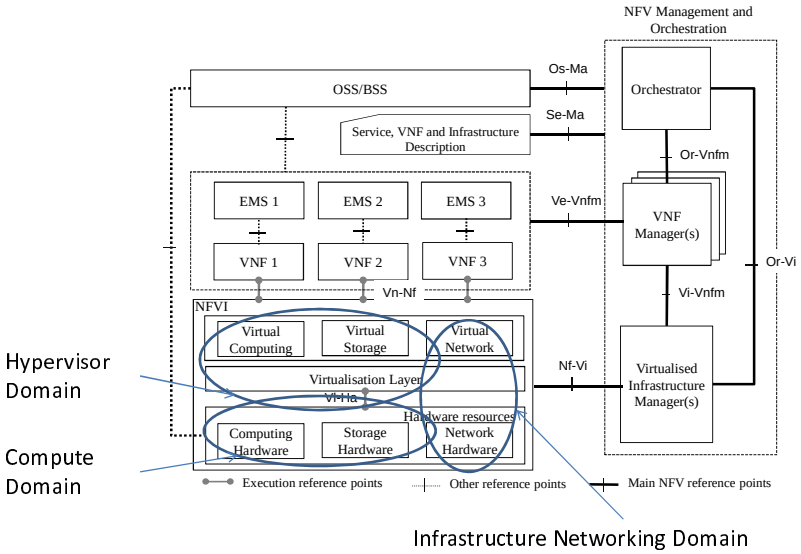

Figure 1: General Domain Architecture and Associated Interfaces

The NFV Infrastructure (NFVI) architecture is primarily concerned with describing the Compute, Hypervisor and Infrastructure domains, and their associated interfaces.

The present document is primarily focused on describing the hypervisor domain, which comprise the hypervisor which:

- provides sufficient abstract of the hardware to provide portability of software appliances;
- allocates the compute domain resources to the software appliance virtual machines;
- provides a management interface to the orchestration and management system which allows for the loading andmonitoring of virtual machines.

Figure 2 depicts the NFV reference architectural framework. Table 1 gives description and definition to the interfaces.

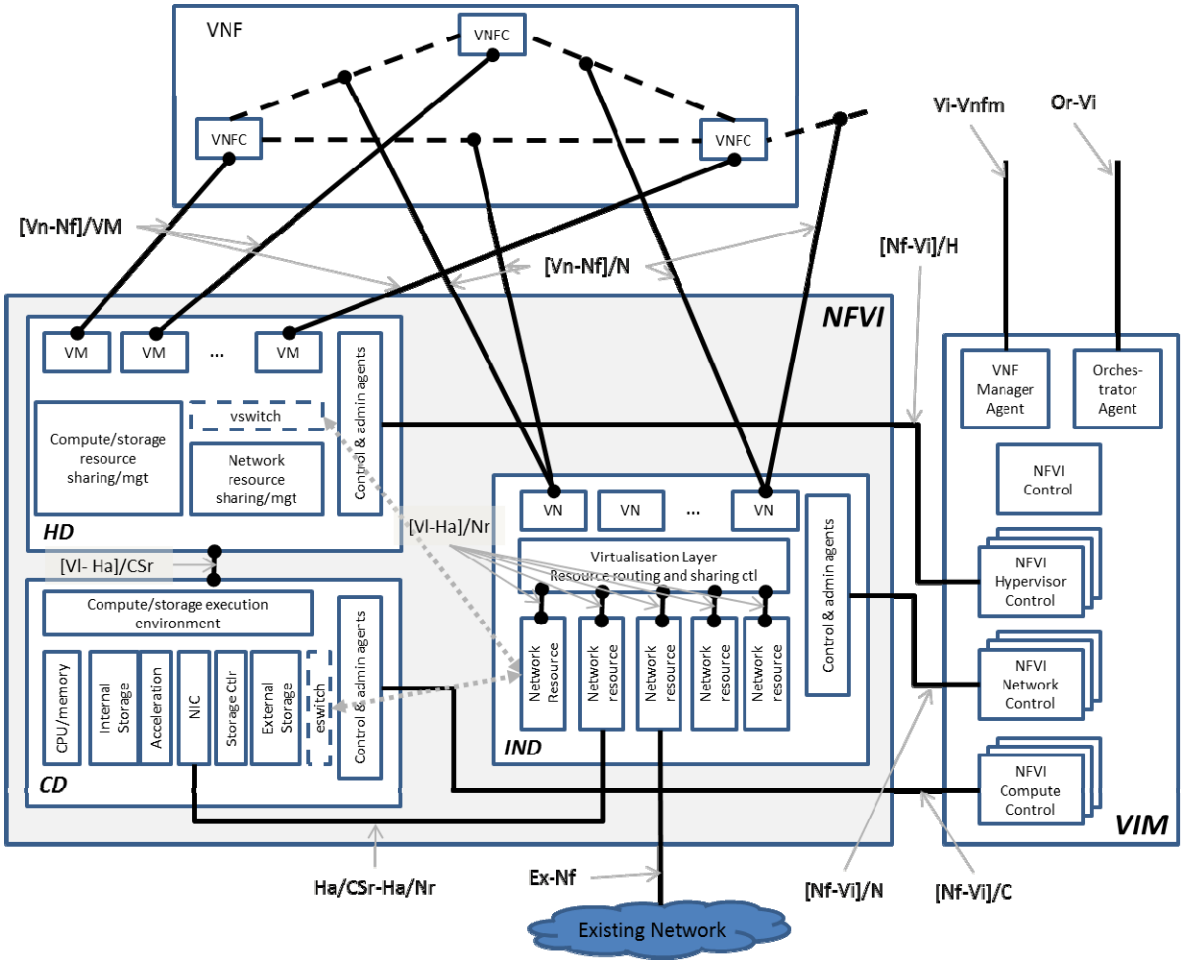

Figure 2: NFV Reference Architectural Framework

Table 1

INF Context	NFV Framework Reference Point	INF Reference Point	Reference Point Type	Description and Comment
External	Vn-Nf	[Vn-Nf]/VM	Execution Environment	This reference point is the virtual machine (VM) container interface which is the execution environment of a single VNFC instance.
		[Vn-Nf]/VN	Execution Environment	This reference point is the virtual network (VN) container interface (eg an E-Line or E-LAN) which carrying communication between VNFC instances. Note that a single VN can support communication between more than a single pairing of VNFC instances (eg an E-LAN VN).
	Nf-Vi	[Nf-Vi]/N	Management, and Orchestration Interface	This is the reference point between the management and orchestration agents in the infrastructure network domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the infrastructure network domain.
		[Nf-Vi]/H	Management, and Orchestration Interface	This is the reference point between the management and orchestration agents in hypervisor domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the hypervisor domain.
		[Nf-Vi]/C	Management, and Orchestration Interface	This is the reference point between the management and orchestration agents in compute domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the compute domain.
	Vi-Vnfm		Management, Interface	This is the reference point that allows the VNF Manager to request and/or for the VIM to report the characteristics, availability, and status of infrastructure resources.
	Or-Vi		Orchestration Interface	This is the reference point that allows the Orchestrator to request resources and VNF instantiations and for the VIM to report the characteristics, availability, and status of infrastructure resources.
		Ex-Nf	Traffic Interface	This is the reference point between the infrastructure network domain and any existing and/or non-virtualised network. This reference point also carries an implicit reference point between VNFs and any existing and/or non-virtualised network.
Internal	Vl-Ha	[Vl-Ha]/CSr	Execution Environment	The framework architecture (see figure 2, NFV Reference Architectural Framework) shows a general reference point between the infrastructure 'hardware' and the virtualisation layer. This reference point is the aspect of this framework reference point presented to hypervisors by the servers and storage of the compute domain. It is the execution environment of the server/storage.
		[Vl-Ha]/Nr	Execution Environment	The framework architecture (see figure 2, NFV Reference Architectural Framework) 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	This is the reference point between the infrastructure network domain and the servers/storage of the compute domain.

The present document focuses on the hypervisor domain. Figures 3 to 5 will depict how the hypervisor domain inter works within the Infrastructure (INF).

The general architecture of a cloud hypervisor is shown in figure 3.

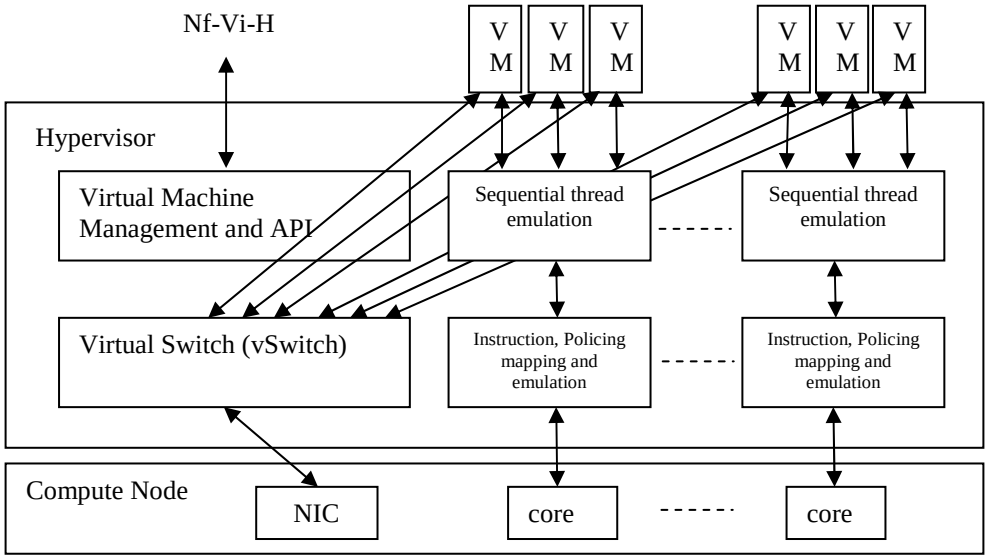

Figure 3

And the architecture of the Hypervisor Domain is shown below.

The hypervisor Domain itself is a software environment which abstracts hardware and implements services, such as starting a VM, terminating a VM, acting on policies, scaling, live migration, and high availability. These services are not instigated without the Virtualisation Infrastructure Manager (VIM) knowing or instructing the hypervisor domainPrimary interfaces of the hypervisor domain:

The NF-Vi interface is the interface to the VIM. This is where the request for hypervisor services occur. Only the VIM or MANO shall interact with the VIM through these interfaces. Hypervisors shall not implement services autonomously unless within the context of the VIM applied policy.

The Vi-Ha interface is the interface that the hypervisor pulls hardware information from and creates virtual hardware components which the Virtual machine utilizes.

The Vn-NF is the VM to VNF logical interface. A VNF is created essentially via one or more Virtual Machines. A virtual machine is in essence software running a function, algorithm, application without being aware of the type, model or number of actual physical units 'underneath' the function, algorithm and/or application.

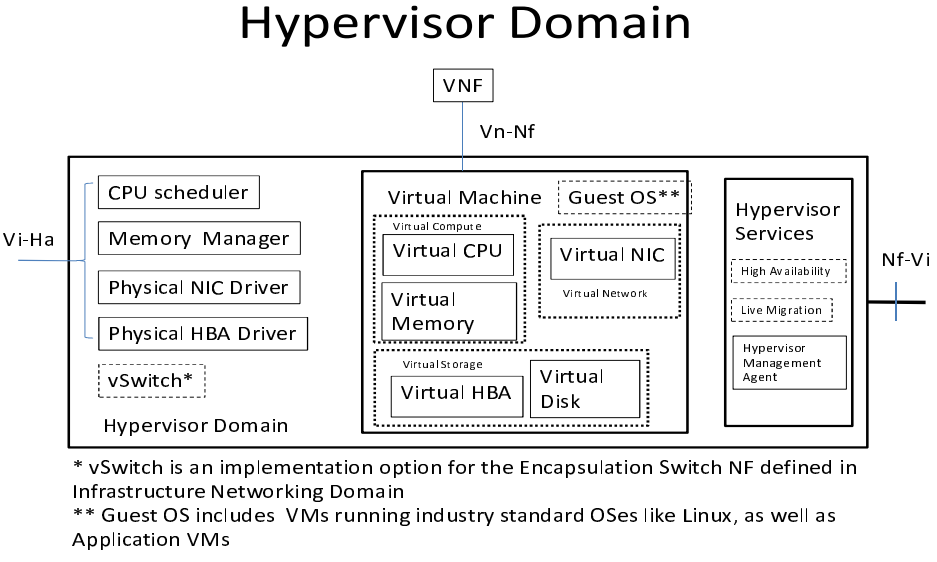

Figure 4

Application VMs definition: is a VM not utilizing an OS:

- server hardware may provide a number of capabilities that enhance virtual machine (VM) performance including:multicore processors supporting multiple independent parallel threads of execution;
- specific CPU enhancements/instructions to control memory allocation and direct access on I/O devices to VM memory allocations;
- PCI-e bus enhancements, notably Single Root I/O virtualisation (SR-IOV).

Hypervisor support for high performance NFV VMs include:

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

The resulting hypervisor architecture is one the primary foundations of the NFV infrastructure. The NFV hypervisor architecture is shown in figure 5.

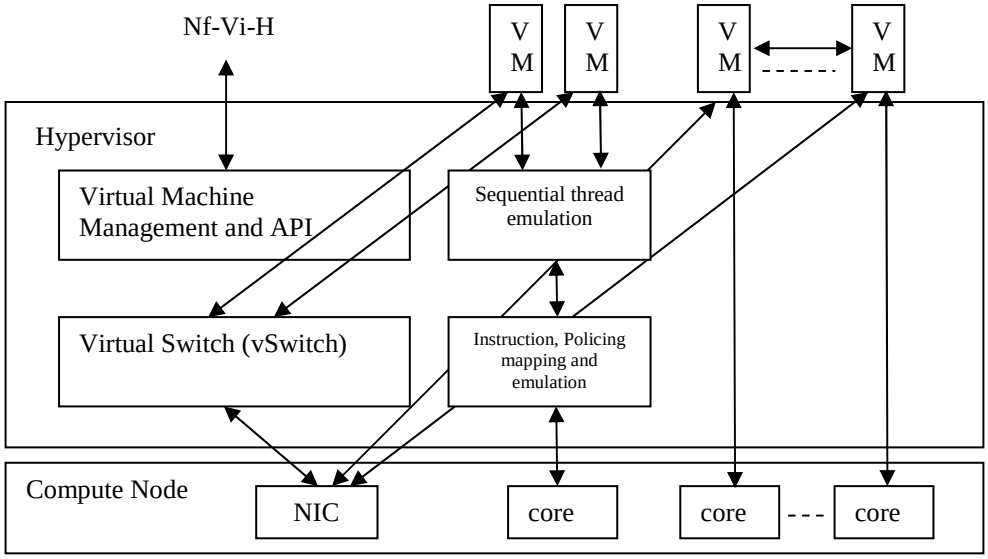

Figure 5

Figure 5 shows the NFV hypervisor architecture and is largely defining of the overall NFV architecture. It is one of a relatively small number of critical components which enable the objectives of NFV to be met. A possible interim and non-preferred alternative, which can still provide some of the benefits of NFV, is to accept a significantly reduced modularly of hardware resource as being the server itself and then dedicate a whole server to a software appliance module. This would also rely on the server to provide the means of remote install and management. While most servers do provide such an interface, they are proprietary and these are not the primary focus of orchestration and management systems. However, such an arrangement could still offer significant advantages over custom hardware and also provide and a reasonable migration path to the full NFV architecture with NFV hypervisors.

5 External Interfaces of the Domain

This clause is a catalogue of the external interfaces of the domain. Given the purpose and definition of the domains includes representing practical boundaries of supply, these are the key interfaces and the ones which need to achieve interoperability of one form or another.

The list of interfaces should follow directly from the first picture in the domain overview.

Given these are the key interfaces, I think it would be sensible to follow a common list of sub-sections:

- Nature of the Interface - given the breadth of convergence the *NFVi Hypervisor Domain WG* are dealing with, there is a very wide diversity of interface type and way interfaces are specified. This sub-section should clarify the nature of the interface which may be obvious to those from that immediate industry sector, but may well not be obvious to many others involved with NFV.
- Specifications in Current Widespread Use - this should be a list of interfaces in actual use and is likely to contain interfaces which have at least proprietary nature to them.
- Achieving Interoperability - the practical means used/needed to achieve interoperability.
- Recommendations - this could be recommending specific existing standards, recommending changes to existing standards, recommending new standards work, recommending that pragmaitic bi-lateral development to interoperate actual implementations is good enough, etc. in the hypervisor document the interfaces are described in the overview section of the document.

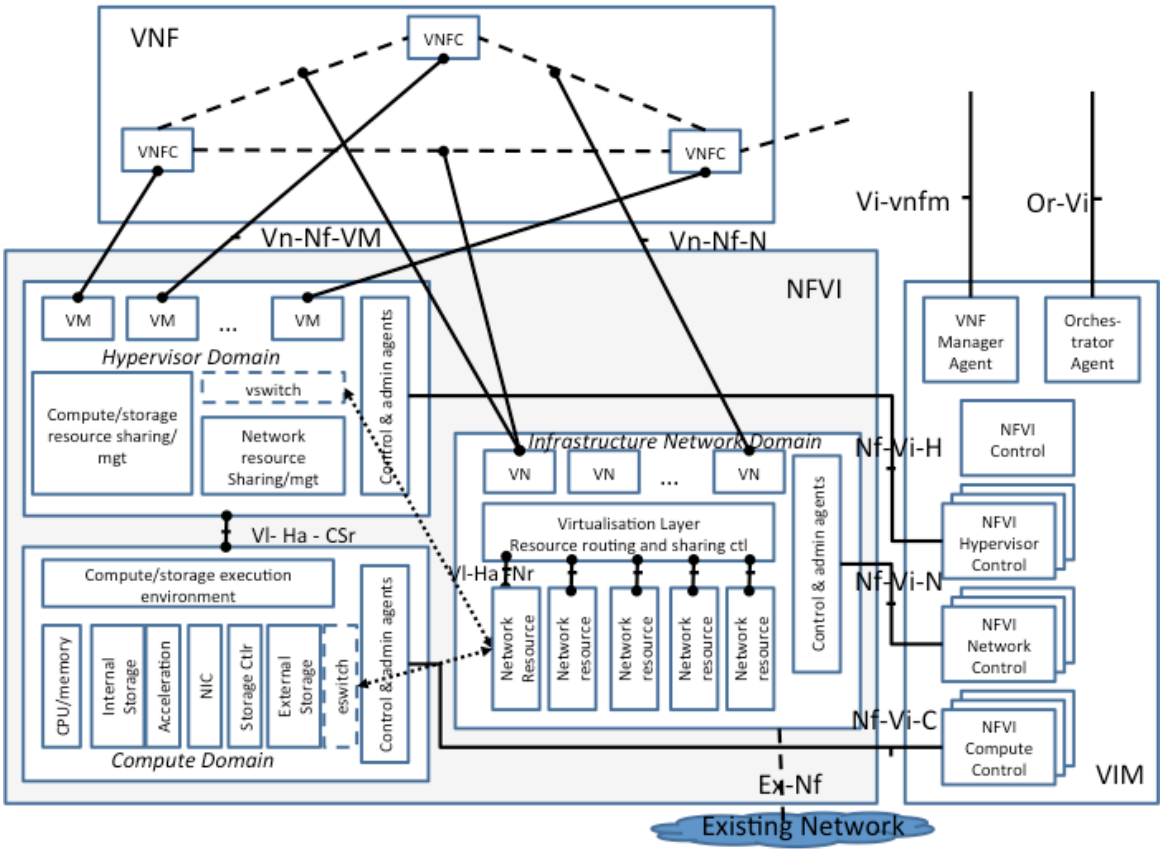

Figure 6

5.1 Overview

The VI-HA-CSr is the interface between the hypervisor and the compute domain. This interface serves the purpose of the hypervisor control of the hardware. This interface enables the abstraction of the hardware, BIOS, Drivers, I/O (NICs), Accelerators, and Memory.

In order for the information model to get created a standard is in place in IETF [i.5]: http://tools.ietf.org/html/draft-ietf-opsawg-vmm-mib-00.

Ext /H.1: hypervisor shall gather all relevant metrics from the compute domain and provide data to the VIM.

The VI-HA-Nr is the interface between the hypervisor and the network domain.

Ext /H.2: hypervisor shall gather all relevant metrics from the networking domain and provide data to the VIM.

Ext/H.3: The hypervisor shall abstract the hardware and create virtual hardware, and provide appropriate level of separation between VMs and VNFs.

5.2 Hypervisor to VIM (Nf-Vi-H) Interface

The Nf-Vi-H is the interface between the hypervisor and the Virtualisation Infrastructure Manager (VIM). This interface serves multiple purposes below is a description of each purpose:

1) The hypervisor sends monitoring information to the VIM, of the underlying infrastructure. This is currently done through various vendor specific packages. It is a requirement of the VIM to utilize the current vendor specific packages. There may be a gap in this interface with respect to a common standard API requirement in order for the VIM to be able to access various different hypervisor schemes and extend the requirements of the interface.A common standard hypervisor monitoring API has yet to be defined and represents a gap.
There are software packages available to implement across different hypervisors. However research is needed and input from across NFV working groups on the gaps with regard to a standard API,what information is transferred (are all the metrics covered) and how the information is transferred (CIM, SNMP, etc.).

2) The VIM is the sole hypervisor controller. All necessary commands, configurations, alerts, policies, responses and updates go through this interface.

5.2.1 Nature of the Interface

The nature of this interface is an informational model. There are informational models supporting the data communication between the virtualisation layer and the virtualisation infrastructure manager (VIM) in deployment today. Vmware, Citrix, Redhat, Wind River System., Debian, CentOS all have a VIM. Openstack potentially could be used to be the framework of a VIM that would utilize any VIM thru a standard method.

Currently there are software products on the market today that interoperate with the various VIMs in the market place:

As an example, one such product is Hotlink: http://www.virtualizationpractice.com/hotlink-supervisor-vcenter-for-hyper-v-kvm-and-xenserver-15369/.

It is a recommendation that there is a standard, or standard opensource that can be a VIM to interwork with multiple commercial VIMs in deployment today in order to not re-write, re- create current VIMs.

Below is a starting point for discussion with regards to virtualisation, BIOS, hypervisors, firmware, networking and hardware.

Ultimately there are levels of Metrics, from high level KQIs: (how long does it take to start up the system, how long does it take to delete a system, etc.); all the way down to the hardware or compute domain. Or alternatively from the compute domain, there are hardware capabilities exposed into the software domain via registers, bios, OS, IPMI, drivers, up thru the virtualisation domain, which runs algorithms sent into the VIM for further calculations and from the VIM to the NFV Orchestrator for additional calculations to get to the evaluation of KQIs.Information models are used along the way to gather and communicate these metrics, results and performance.

The next section gives examples of data contained in an information model, followed up by a section that gets into what the hardware provides to software in order for the information model to get the data into feature designed to calculate SLA performance criteria and requirements.

Neither of these two following sections are anywhere close to be exhaustive. Thus, the need for an NFV WI to research what is required above and beyond what is available today.

5.2.1.1 Example of MIB information

There are over 1 000 parameters/variables/metrics that are used in Virtualisation/ Cloud Service Assurance MIBs.

Hypervisors and cloud programs use IDs, Bios, IPMI, PCI, I/O Adapters/Drivers, memory subsystems, etc. to get to all of the items of current concerns, including information for failover, live migration, placement of the VMs/applications.

It is not clear that there is a gap at this point. The hypervisor team has indicated the exposure of the hardware capabilities thru what is in the documentation today (id info, impi, drivers, bios, pci, memory subsystems, etc.) has not exposed any gaps. The hypervisor team is looking forward to working with the Metrics WI, SWA, SEC and MANO for any gaps or requirements beyond what is available today.

The NFVINF hypervisor domain VIM is expected to leverage available managers such as CloudStack, vCenter, Openstack, others as packages. There are software packages available today that implement this scheme, e.g. HotLink SuperVisor: editor's note to scrub for trademarks.

http://www.virtualizationpractice.com/hotlink-supervisor-vcenter-for-hyper-v-kvm-and-xenserver-15369/

Below is a table of some of the components currently in a MIB, informational model, that show some of the functions.

MIBs are generally broken up into 'functional' areas. Below are some examples.

MIB Functions/objects:

- Resources (CPU, Memory, Storage, Adapters, Resource pools, Clusters):
  - Ex: Adapters: PCI ID, I/O, memory, bus, model, status, capabilities.
- Systems (Logs, NUMA, I/O, etc.).
- Events.
- VM management.
- Obsolete/Legacy (compatibility with older versions).
- Products (supported products (hw and sw)).
- Analytics:
  - Checks the incoming metrics for abnormalities in real time, updates health scores, and generates alerts when necessary.
  - Collects metrics and computes derived metrics.
  - Stores the collected metrics statistics. (filesystem).
  - Stores all other data collected, including objects, relationships, events,dynamic thresholds and alerts.
- Multicore Processors:
  - Hyperthreading.
  - CPU Affinity's.
  - Power management.

Table 2 contains some details with regard to the VIM information that is gathered. These tables are not complete. Research is required to determine the performance characteristics and the gaps of what is provided today and what is needed.

Table 2

Functional objects	Data exposed to VIM	
Main System (Base Server) System	System temperature exceeded normal operating range	Individual and overall temperature sensor health status, including temperature readings
	System temperature has returned to normal operating range	System manufacturer, model, BIOS version and date
	Server model, serial number, product number and universal unique identifier (UUID)	System OS name, type, version number and description
	System up time	Monitoring subsystems such as IPMI
	Sensor types	One entry is created for each physical component
	Computer System Consolidated health status	Monitoring information from Drivers, BIOS, OS
Processor (CPU) Subsystem	Individual processor number, core and thread number, speed, physical socket location and health status	The number of physical cores in the system
	Individual processor cache size, line size, cache level and type, read and write policy and health status	Individual and overall processor health status
	Individual processor chip model, manufacturer, version	Individual processor model, speed, sockets, cores, logical processors
	Processor temperature thresholds and crossings	Processor temperature has returned to or exceeded normal operating range

Table 3

Functional objects	Data exposed to VIM	
Power Supply Subsystem	Individual power supply type, physical power supply location and health status	Individual and overall power supply health status
	Power supply redundancy set, number of power supplies, associations with individual power supply members, and redundancy status	
	Individual power supply module removal conditions and package type	Power supply temperature exceeded normal operating range
	Power supply collection health status	Power supply temperature returned to normal operating range
Memory Subsystem	System memory capacity, starting and ending address, and health status	Memory module has failed or is predicted to fail
	Individual memory module manufacturer, part number, serial number, removal conditions, data and total width, capacity, speed, type, position, form factor, bank label, location and health status	Memory board error
	Individual memory board package type, removal conditions, hosting board, locked state	Memory redundancy degraded
	Number of sockets, available memory size, total memory size, location and health status	Memory recovered from degraded redundancy
	Individual memory module slot connector layout, gender and description, location, and health status	Amount of physical memory present on the host
	Memory redundancy set type, load balance algorithm, operating speed, available and total memory size, current, target and available configurations, and redundancy status	Amount of physical memory allocated to the service console
	Memory collection health status	Amount of memory available to run virtual machines and allocated to the hypervisor

Table 4

Functional objects	Data exposed to VIM	
Power system/sub-system	Power usage/capacity	Redundant Power System Units

Some of the metrics are calculated by doing processing on the metrics provided by the hardware.

5.2.2 Specifications in Current Widespread Issue

KVM libvirt, VMware vSphere, XenServer XAPI for Vf-Vi-H VIMs.

Internal interface: For VI-HA-CSr: see the compute domain for hardware interfaces.

5.2.2.1 Metrics for VNF performance characteristics

Below are some examples of metrics that will be provided by the compute domain in order for an orchestrator to be able to determine the status, nature of the network. This is not an all inclusive list. See the NFVINF Compute Domain GS for further information.

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 VIM/hypervisor could proactively update hypervisor and software domains at regular intervals based on policies.

Annex A has a more exhaustive list of metrics and VIM categories that is exposed and implemented by the hypervisor.

Table 5: General Example Table of Metrics gathered by the hypervisor

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 following dynamic metrics (an initial proposal) to quantify this function.

Table 6: Dynamic hardware 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 an VNFi.* • Is this wording sufficiently accurate?).
	Failed Initializations	• Count of attempted Initializations that do not result in the server/blade reaching the state of readiness for commissioning.
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? Better example?).
	Failed Activations	• Count of attempted Activations that do not result in the storage unit reaching the state of readiness.

Upon examine the 3x3 matrix for CPU-related dynamic metrics, there are the following coverage of functions and criteria.

Table 7: Dynamic hardware Activation 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			

Internal interface: For VI-HA-Nr: see the networking domain.

5.2.3 Achieving Interoperability

From the interface point of view, there are multiple 'packages' (vCenter, Cloudstack, etc.) that MANO needs to interface and/or translate, and the VIM needs to manage.

5.2.4 Recommendations

VIM needs to manage multiple hypervisors from multiple vendors.

6 Architecture and Functional Blocks of the Hypervisor Domain

This clause is a catalogue of the functional blocks of the domain. The primary purpose of setting out functional blocks is to link with the cross domain aspects of management and orchestration, performance, reliability and security.

With this objective, the delineation of functional blocks cannot be completely independent of implementation, as some aspects of implementation affect all of these cross domain areas to a greater or lesser extent.

There are different configurations available that will be discussed in this section and referenced in the requirement section.

A picture of Direct Assignment and SR-IOV is shown below to give the reader a sense of what the configuration entails.

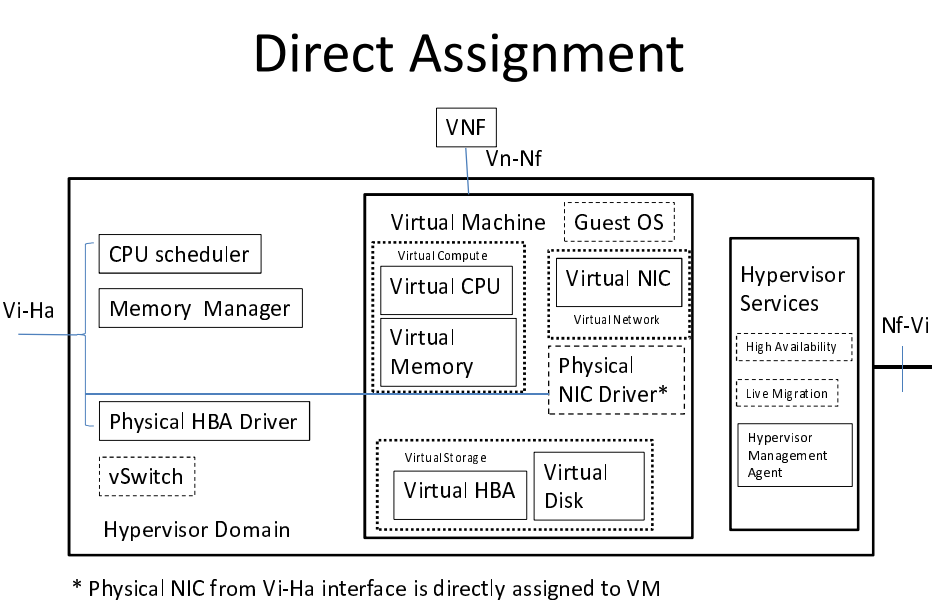

Figure 7

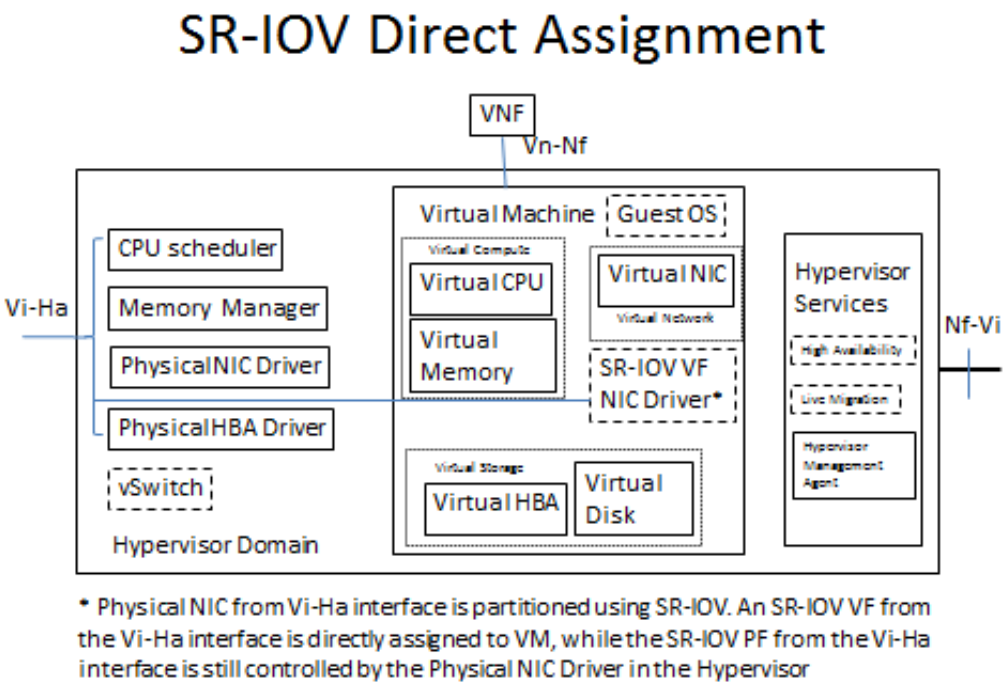

Figure 8

7 Requirements for the Hypervisor Domain

This clause of the document links to the ISG Level Requirements. In an effort to focus on these requirements and the industry gaps, the document has been restructured to specifically call out the ISG level requirements and the detailed hypervisor requirements supporting the ISG level requirement follow each requirement that impacts the hypervisor domain.

ISG Level doc name ETSI GS NFV 004 [i.1].

There is a transition stage in the industry and the tendancy is to focus on performance based on a specific instance. It is the goal of the NFV to move towards not to highly performant monolithic VMs but performance thru distribution of function and capability across multiple VMs and VNFCIs. Scale and speed to market is the goal, not optimizing every instance.

In Brackets **[Bold**] is the ISG level requirement. Any associated hypervisor requriements necessary to support meeting the ISG level reguirement will follow with reguirments called out with the same ISG level requirement name, no brackets with and addition of \H.xx where xx will be a number.

EXAMPLE 1: **[Gen.1], [Gen.2]…** are ISG level requirements.

EXAMPLE 2: **Gen\H.1** is one of the hypervisor requirement that is required to support the ISG level requirment.

7.1 General

[**Gen.1**] Service providers shall be able to *partially* or *fully* virtualise the network functions needed to create, deploy and operate the services they provide.

[**Gen.2**] In case of partial virtualisation, there shall only be manageable, predictable and acceptable impact on the performance and operations of the network functions that have not been virtualised.

[**Gen.3**] In case of partial virtualisation, there shall be manageable impact on the legacy management systems of the network functions that have not been virtualised.

[**Gen.4**] The NFV framework shall be able to support a service composed of physical Network Functions (NFs) and virtual Network Functions (NFs) implemented across data centre multivendor environments that may be instantiated in a single operator or in cooperating inter-operator environments. The following areas are in or out of scope.

Table 8

Scenario\NF Composition	All Physical NFs	Physical/Virtual NFs	All Virtual NFs
Intra-Operator	Out-of-Scope	In-Scope	In-Scope
Inter-Operator Cooperation	Out-of-Scope	In-Scope	In-Scope

7.2 Portability

This clause will focus on the ability to move VMs. The initial section restates the high level ISG Virtualisation Requirements that the hypervisor requirements are linked to. Following the ISG requirements will be the requirements of the hypervisor in order to meet the portability requirements.

ISG Level Requirements

[**Port.1**]**:** The NFV framework shall be able to provide the capability to load, execute and move virtual machines and/or appliances across different but standard data centre multivendor environments.

Note that this is an **industry gap**: the ability to move hypervisor-level VMs between hypervisors and CPU architectures is not fully supported, and any likely technical solutions will introduce performance impact.

Hypervisor Requirement

Port/H.1: In order for the ability to move an application/VM/appliance across standard multivendor environment, the VNF shall be able to run on any standard hypervisorCurrently Portability between silicon and portability between hypervisors is an industry gap.

Port/H.2: The Vn-NF shall have/be an API that is consistent and useable across any standard hypervisor. It is anticipated that this is a software architectural gap.

Port/H.3: A mechanism shall be in place to allow communications between the hypervisor and the VM.

Port/H.4: Hypervisor shall be able to deny requests.

Security and other issues may influence decision.

Port/H.5: A mechanism shall be in place to allow a VM to be notified that it is going to be moved.

Port/H.6: Configuration of VM to VM communication; intra- and inter-host; shall have standard or open interfaces.

Port/H.7: Configuration of network connectivity services; intra- and inter-host; shall have standard or open interfaces. This is covered in [insert ref to network domain]. This could require a VM working on one type of hypervisor to communicate to another VM on a different hypervisor. This use case exposes a gap in the software architecture and potentially in MANO.

GAP: Business drivers and technology to enable across Hypervisor live migration.

GAP: Business drivers and technology to enable across ARM & Intel live migration.

GAP: Image Package format interoperability is a gap between the hypervisor providers.

[**Port.2**]**:** The NFV framework shall define an interface to decouple software instances from the underlying infrastructure (as represented by virtual machines and their hypervisors).

Hypervisor Requirements

Port/H.8: The hypervisor shall be able to unbind the VM from the hardware in order to allow the VM to be portable.

Port/H.9: The hypervisor shall provide a mechanism for VM migration between hosts.

Port/H.10: The hypervisor shall provide a mechanism for VM migration between hosts which ensures atomicity of VM instances.

Port/H.11: The hypervisor shall provide metrics to allow the management and orchestration entities to make predictions as to the impact of migration.

[**Port.3**]**:** The NFV framework shall be able to provide the capability to optimize the location and required resources of the virtual appliances.

Hypervisor Requirement

Port/H.12: The hypervisor software shall be able to move the VMs.

Port/H.13: The hypervisor shall have minimal impact on network performance. The hypervisor shall be able to recognize the existence of features and functionality provided by resources such as the compute, storage and networking and may make use of them to optimise VM performance.

Port/H.14: Where standards for functions and capabilities exist, they shall be employed, rather than proprietary or closed methods.

The SR-IOV specification, for example, provides a standard mechanism for devices to advertise their ability to be simultaneously shared among multiple virtual machines. It also allows for the partitioning of a PCI function into many virtual interfaces for the purpose of sharing the resources of a PCI Express device in a virtual environment.

Port/H.15: Functions and capabilities such as SR-IOV, vswitch, performance, tunnelling protocols, etc. shall be exposed to the management and orchestration entities using standard APIs which need to be aligned with the Network Infrastructure Domain.

Port/H.16: Moving VMs requires the same instruction set in which the application was built on.The same instruction set enables the movement of applications between systems. Different instruction sets will have to be accommodated by application aware or extra software when moving applications or VMs.

Port/H.17: The hypervisor exposes to MANO, via the VIM, accelerator information and capabilities.

Port/H.18: The hypervisor exposes to MANO, via the VIM, accelerator information and capabilities employed by a VM.

Port/H.19: The hypervisor shall have the ability to scale a VM up and down: to add/remove compute and memory dynamically and shall expose such capabiltities to MANO.

Port/H.2: The hypervisor shall have the ability to use static resourcing during maintenence or other modes deemed necessary by the operator dynamically and shall expose such capabiltities to MANO.

Port/H.21: The hypervisor shall have the ability to put resources into a lower power state based on utilization or a control message dynamically and shall expose such capabiltities to MANO.

Port/H.22: The hypervisor shall be able to discover processor identification and feature set and to allocate CPU resources to a VM which are appropriate for the needs of that VM in terms of CPU family, features and capabilities, for example, Intel's cpu id instruction.

NOTE: The ability to virtualise the instruction enables the control over the values reported by instructions to the guest software.

Port/H.23: Processor family and capability information shall be made available to MANO to allow correct placement of VMs which have specific requirements in these areas.

Port/H.24: A hypervisor shall support the virtualisation of the processor ID.

Port/H.25: A hypervisor shall support the live migration of VMs between compatible processors that support differing feature sets.

Port/H.26: A hypervisor shall report a processor ID other than that reported by the compute node when the feature set of the compute node processor is compatible with the feature set of the processor ID reported by the hypervisor.

Port/H.27: A hypervisor shall make use of the available hardware assistance to virtualise, partition and allocate physical memory among the VMs.

Port/H.28: A hypervisor shall make use of the available hardware assistance to re-map Direct Memory Access (DMA) transfers by I/O devices to direct accesses to the allocated memory of the client VMs of the I/O device.

Port/H.29: A hypervisor shall make use of the available hardware assistance to re-map device generated interrupts to direct them to client VMs of the I/O device.

Port/H.30: DMA remapping shall be utilized that allows for reduction in VM exits for assigned devices.

Port/H.31: Interrupt remapping shall be enabled to allow for reductions in interrupt virtualisation overhead for assigned devices.

Port/H.32: Hardware page virtualisation shall be utilized to improve performance. Performance benefits from hardware page virtualisation are tied to the prevalence of VM exit transitions. Hardware assisted extended page table allows a guest OS to modify its own page tables and directly handle page faults. This avoids VM exits associated with page-table virtualisation.

Port/H.33: The hypervisor domain shall enable direct access to processor cache in order to facilitate performance.

7.3 Elasticity/Scaling

The following requirements apply when components within VNF are capable of simultaneous, parallel operation in more Virtual Machine instances:

[**Elas.1**]**:** The VNF vendor shall describe in a 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.2**]**:** The NFV framework shall be able to provide the necessary mechanisms to allow virtualised network functions to be scaled. Three mechanisms shall exist: manual, on-demand or automatically. On-demand scaling of a VNF instance may be initiated by the VNF instance itself, by another authorized entity (e.g. OSS), or by an authorized user (e.g. a NOC administrator).

[**Elas.3**]**:** The scaling request or automatic decision may be granted or denied depending on e.g. network wide views, rules, policies, resource constraints or external inputs.

[**Elas.4**]**:** The licensed VNF user shall be capable of describing in a information model for each component capable of parallel operation the minimum and maximum range of such VM instances for which it will request licenses. These limits shall be at lest the minimum and no greater than the maximum specified by the VNF vendor.

Hypervisor Requirements

Elas/H.1: The hypervisor shall make available to MANO information as to resources available for scaling.

MANO communication for scale up/down: adding and removing resources into/from a VM.

Elas/H.2: Mano makes the request thru the VIM; hypervisor shall attempt to fulfill; the result will be available to the VIM; All requests are mediated via MANO.

Software and MANO: VNFs shall be aware of their state and communicate to MANO its current resource allowing MANO to request hypervisor to scale in/out: (the initiate or termination of Vms).

Elas/H.3: Hypervisor shall attempt to fulfill request from the VIM to create or terminate VMs; the result will be available to the VIM; All requests are mediated via MANO.

For a single hypervisor domain of cpu & meory scaling is limited to cache coherent resources today.

No requirements are derived from the Elas.

7.4 Resiliency

[**Res.1**]**:** The NFV framework shall be able to provide the necessary mechanisms to allow network functions to be recreated after a failure.

NOTE 1: Two mechanisms exist: on-demand or automatically.

Reference is made to the technical challenges clause on Security and Resiliency from the White Paper "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1".

[**Res.2**]**:** The NFV framework shall be able to provide a means to classify (sets of) VNFs that have similar reliability/availability requirements into resiliency categories.

[**Res.3**]**:** A (set of) VNF instance(s) and/or a management system shall be able to detect the failure of such VNF instance(s) and/or network reachability to that (set of) VNF instance(s) and take action in a way that meets the fault detection and restoration time objective of that VNF resiliency category.

Requirement: There is a requirement on liveness checking of an VNF should be described - e.g. watchdog timer or keepalive at the NF level. Means includes interface, syntax, methods for discovering, publishing, and retrieving notifications. Need to include requirements of the interaction between the VNF (image) and the VM manager (hypervisor). Storage/file and networking locks/synchronization need to be covered in RELAV.

[**Res.4**]**:** A VNF shall be able to publish means by which other entities (e.g. another VNF, the ality and/or a management system) can determine whether the VNF is operating properly.

[**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 fulfil the SLAs.

NOTE 2: "Standard" includes de-jure, de-facto or open standard that fulfils the requirement. This assumption should be applied to all requirements through the entire document.

[**Res.6]:** The NFV framework (including the orchestration and other functions necessary for service continuity) shall not include a single point of failure (SPoF).

[**Res.7]:** The SLA shall specify the "metrics" to define the value and variability of "stability".

[**Res.8**]**:** The NFV framework shall guarantee network stability using all or some the following metrics:

- Maximum non-intentional packet loss rate (i.e. packets lost due to oversubscription of the service network interconnects, not due to policers or filters).
- Maximum rate of non-intentional drops of stable calls or sessions (depending on the service).
- Maximum latency and delay variation on a per-flow basis.
- Maximum time to detect and recover from faults aligned with the service continuity requirements (zero impact or some measurable impact).
- Maximum failure rate of transactions that are valid and not made invalid by other transactions.

NOTE 3: Metrics to define the value and variability of "network stability" are required and additional metrics may be added to the above list. Changes in topology, or next hop changes (on a per flow or per target basis) are not useful metrics as inserting virtualised functions does not necessarily change those. How to measure packet loss, delay and delay variation while the VNFs are implemented is FFS.

7.5 Security

[**Sec.1**]**:** The NFV framework shall be able to ensure the security of the infrastructure, especially the hypervisor.

Managing Access to Restricted Resources.

Enterprise administration often requires security and management agents to be placed on user machines. It is desirable to make these inaccessible both to users and to unauthorized code. For example, restricting access to an intrusion-detection agent in this way could prevent it from being removed by the user or compromised by malicious code.

Sec/H.1: A VM on a hypervisor shall not be able to compromise the hypervisor or other VMs.

Sec/H.2: Hypervisor will ensure that only authorized VMs will be allowed access to resources, based on policies provided by MANO.

Security requirements shall come from Security WG.

[**Sec.2**]**:** The NFV framework shall be able to provide mechanisms for the network operator to control and verify hypervisor configurations.

[**Sec.3**]**:** Orchestration functionalities shall be able to use standard security mechanisms wherever applicable for authentication, authorization, encryption and validation.

[**Sec.4**]**:** NFV Infrastructure shall be able to use standard security mechanisms wherever applicable for authentication, authorization, encryption and validation.

Reference is made to the technical challenges clause on Security and Resiliency from the White Paper "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1".

Introspection Risk

Administrative and process introspection presents a risk to the confidentiality, integrity and availability of the NFV ecosystem. Introspection is the ability to view and/or modify operational state information associated with NFV through direct or indirect methods. Access to state information can result in the ability to arbitrarily read and/or write the contents of memory, storage, keystores and other NFV operational aspects. Access can be direct via CLI (command line interfaces), GUI (graphical user interfaces), DMA (direct memory access), API (application programming interfaces) and other methods. Indirect access involves the use of sidechannels and other means that allow the inference of data and modification via resources with privileged relationships.

Administrative models that enable an admin, root or superuser account type that has full access to system resources allows visibility into and modification of private keys, symmetric keys, passwords in memory, networking, system configuration and other trusted resources. Modification of logging, reporting and alerting can disguise malicious access, unintentional modification, vulnerabilities and administrative mistakes Likewise, similarly privileged processes posses these capabilities.

The use of administrative models and process introspection that could allow unfettered access and modification capabilities should be clearly documented as an architectural, operational and audit risk to the NFV ecosystem.

7.6 Service Continuity

The NFV framework shall provide the following capabilities:

[**Cont. 1**]**:** The SLA shall describe the level of service continuity required (e.g. seamless, non-seamless according to the definitions) and required attributes.

[**Cont. 2**]**:** In the event of an anomaly, which causes hardware failure or resource shortage/outage, the NFV framework shall be able to provide mechanisms such that the functionality of impacted vNF(s) shall be restored within the service continuity SLA requirements for the impacted VNF(s).

[**Cont. 3**]**:** In the event that a vNF component needs to be migrated, the NFV framework shall be able to consider the impact on the service continuity during the VNF migration process and such impact shall be measurable and within the limits described in the SLA.

[**Cont. 4**]**:** When a vNF component (e.g. a VM) is migrated, the communication between the vNF and other entities (e.g. VNF components or physical network elements) shall be maintained regardless of its location and awareness of migration.

Once many network functions are provided in virtualised form, the more ubiquitous presence of virtualisation infrastructure presents the opportunity for running virtualised instrumentation functions whenever and wherever they are needed, e.g. to remotely view a point in the network to diagnose problems, to routinely monitor performance as seen by a customer or to measure latency between two remote points. Most instrumentation solutions (whether hardware or virtualised) work to a common pattern, where packets or headers of interest are copied and time-stamped, then forwarded to a function that then analyses them out of band by reconstructing the sequence of events using the time-stamps. Therefore sufficiently precise time-stamping will be needed in any circumstances where the relation between one function and another may be investigated, including when replica functions are created or functions are migrated.

[**Cont.5**]**:** It is believed that time-stamping frames in virtualised software may be insufficiently precise for many instrumentation purposes. Therefore, when tendering for network interface cards (NICs) and network interface devices (NIDs) that sit beneath virtualisation infrastructure, it is likely that network operators will require hardware time-stamping. In such cases, the minimum support from hardware will be to:

- copy packets or frames;
- accurately time-stamp the copies, using a clock synchronised to a source of appropriate precision (e.g. IEEE 1588 [i.6]); 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.

[**Cont.6**]**:** It should be possible to interrogate whether particular network interface hardware provides hardware time-stamping facilities.

NOTE: Detailed requirements and recommendations on how this could be economically implemented should be in scope for study within an appropriate NFV ISG Work Item and/or referenced to a competent body.

Cont/H.1: The hypervisor shall provide the ability to perform measurements of the frequency and extent of VM stall events, and conduct the measurements and report the occurrence and extent of each stall event on VMs where this capability has been activated.

7.7 Operational and Management requirements

[**OaM.1**]**:** The NFV framework shall incorporate mechanisms for automation of operational and management functions.

[**OaM.2**]**:** The NFV framework shall be able to provide an orchestration functionality that shall be responsible for the VNFs lifecycle management: install, instantiate, allocate resources, scale-up/down, and terminate.

[**OaM.3**]**:** The orchestration functionality shall be limited to the differences introduced by the Network Function Virtualisation process. The orchestration functionality shall be neutral with respect to the logical functions provided by the VNFs.

NOTE 1: Operational, Provisioning or Management aspects of the logical functions of the VNF are not considered part of the orchestration functionality depicted in the present document.

[**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 describe how to manage the VNF life cycle. Information models describe the deployment and operational attributes of VNFs.

For example, hypervisors do not tear down VNf, or migrate VNF:

- deployment attributes and environment of a VNF e.g. VM images, required computational and storage resources and network reachability;
- operational attributes of a VNF, e.g. VNF topology, operations (initiation/tear-down), functional scripts.

NOTE 2: The examples above are not exhaustive and can be refined during further phases of the NFV work.

[OaM.6]: The orchestration functionality shall be able to manage the lifecycle of a VNF using the information models in combination with run-time information accompanying automated or on-demand requests regarding VNF instances and run-time policies/constraints.

NOTE 3: The orchestration function will include tools to measure a VNF's instance service availability and reliability against requirements supplied in the template of the VNF(s) it manages.

[**OaM.7**]**:** The orchestration functionality shall be able to manage the NFV infrastructure in coordination with other applicable management systems (e.g. CMS) and orchestrate the allocation of resources needed by the VNFs.

[**OaM.8**]**:** The orchestration functionality shall be able to maintain the integrity of each VNF with respect to its allocated NFV infrastructure resources.

[**OaM.9**]**:** The orchestration functionality shall be able to monitor and collect NFV infrastructure resource usage and map such usage against the corresponding particular VNF instances.

[**OaM.10**]**:** The orchestration functionality shall be able to monitor resources used by VNFs, and shall be made aware of receiving events that reflect NFV Infrastructure faults, correlate such events with other VNF related information, and act accordingly on the NFV Infrastructure that supports the VNF.

[**OaM.11]:** The orchestration functionality shall support standard APIs for all applicable functions (e.g. VNF instantiation, VNF instances allocation/release of NFV infrastructure resources, VNF instances scaling, VNF instances termination, and policy management) that it provides to other authorized entities (e.g. OSS, VNF instances, 3rd parties).

[**OaM.12**]**:** The orchestration functionality shall be able to manage policies and constraints (e.g. regarding placement of VMs).

[**OaM.13**]**:** The orchestration functionality shall enforce policies and constraints when allocating and/or resolving conflicts regarding NFV Infrastructure resources for VNF instances.

Reference is made to the technical challenges clauses on Automation, Simplicity and Integration from the White Paper "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1".

Life Cycle Hypervisor Requirements: see Scaling section.

Overview:

Hypervisor Requirements

OaM/H.1: The hypervisor shall make available to MANO information as to resources available.

OaM/H.2: Mano makes the request thru the VIM to initiate or terminate a VNF; hypervisor shall attempt to fulfill the request; the result will be available to the VIM; All requests are mediated via MANO.

Software and MANO: VNFsshall be aware of their state and communicate to MANO its current resource allowing MANO to request hypervisor to scale in/out: (the initiate or termination of Vms).

OaM/H.3: hypervisor shall attempt to fulfill request from the VIM to create or terminate VMs; the result will be available to the VIM; All requests are mediated via MANO.

OaM/H.4: For a single hypervisor domain of cpu & memory, scaling is limited to cache coherent resources today.

Steps to manage and provision the life cycle of hypervisor are located in annex A.

7.8 Energy Efficiency requirements

Network infrastructures consume significant amounts of energy. Studies have indicated that NFV could potentially deliver up to 50 % energy savings compared with traditional appliance based network infrastructures. The virtualisation aspect of network functions assumes some separation of communication, storage or computing resources, which implies changes in the distribution of energy consumption. VNFs provide on-demand access to a pool of shared resources where the locus of energy consumption for components of the VNF is the virtual machine instance where the VNF is instantiated. It is expected that the NFV framework can exploit the benefits of virtualisation technologies to significantly reduce the energy consumption of large scale network infrastructures.

[**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.

NOTE 1: Workload consolidation can be achieved by e.g. scaling facilities so that traffic load is concentrated on a smaller number of servers during off-peak hours so that all the other servers can be switched off or put into energy saving mode.

Hypervisor Requirements

Using intelligent energy management at the platform level will network operators squeeze extra value and performance out of existing rack space while reducing the total cost of ownership by better managing power and cooling operational costs.

EE/H.1: The hypervisor shall report to the utilization of the resources to the manager/orcastrator in order for manual control to exist for sleep states.

EE/H.2: The hypervisor shall be able to be configured to recognize thresholds of utilization and/or policies to utilize automation to put cpus in a lower power state.

EE/H.3: Going in and out of the lower power states shall ensure the application/vnf meets the SLA.

EE/H.4: The hypervisor shall use power management to utilize policies defined by network management and which improve energy efficiency.

EE/H.5: The hypervisor shall utilize policies that define the different sleep states for processors and have the data on the processors as to the timing or latency of coming out or going into a sleep state in order to meet the SLAs.

EE/H.6: Policies and algorithms are required to take into account processor frequencies and applications to optimize power management schemes: MANO impact.

EE/H.7: ACPI(Advanced Configuration and Power Interface) shall be used along with the OS to work in conjunction to achieve max efficiencies.

[EE.2]: The NFV framework shall be able to provide mechanisms to enable an authorised entity tocontrol and optimise energy consumption on demand, by e.g. scheduling and placing VNFs on specific resources, including hardware and/or hypervisors, placing unused resources in energy saving mode, and managing power states as needed.

NOTE 2: Energy efficiency mechanisms should consider maintaining service continuity requirements and network stability requirements.

EE/H.8: Hypervisor will implement policies provided by MANO, includes power management, power stepping, etc.

[**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.

NOTE 3: This information is necessary to determine when to power on resources and software sufficiently in advance of the time when such assets would be needed to meet expected future workloads.

7.9 Guest RunTime Environment

The virtual machine shall support all major standard Os' including:

- Linux.
- Windows.
- Solaris.

And the virtual machine may support Real time OS, BSD and library OS'.

The guest run time environment may include other run time environments:

- JVM.
- Erlang.
- Python.

SWA Impacts/input required with regard to inpacts to the VNF. There is no reason to suspect that there is an impact.

A Bare Metal OS with a hypervisor is included here for completeness, although clarification is needed.

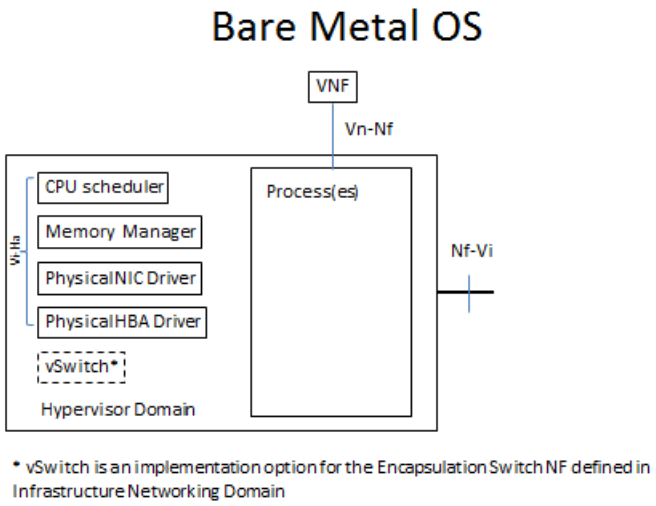

Figure 9

7.10 Coexistence with existing networks - Migration

[**Mig.1**]**:** The NFV framework shall co-exist with legacy network equipment. I.e. the NFV shall work in a hybrid network composed of classical physical network and virtual networkfunctions.

[**Mig.2**]**:** The NFV framework shall support a migration path from today's physical network functions based solutions, to a more open standards based virtual networks functions solutions.

NOTE 1: For example, the NFV migration should allow a true "mix&match" deployment scenario by integrating multiple virtual and physical functions from different vendors without incurring significant integration complexity and facilitate multi-vendor ecosystem.

[**Mig.3**]**:** The NFV framework in conjunction with legacy management system shall support the same service capability and acceptable performance impact when transitioning from physical network functions to virtual network functions (and vice versa).

[**Mig.4**]**:** The NFV framework shall be able to interwork allow with legacy orchestration systems with minimal impact on existing network nodes and interfaces.

NOTE 2: Example of legacy orchestration systems could be cloud management or load balancing control systems that may exist in the operators' networks.

[**Mig.5**]**:** During the transition from physical to virtual NFV framework shall be able to ensure security of VNFs, PNFs and other network elements and interfaces from various attacks.

Reference is made to the technical challenges clause on Coexistence with existing networks - migration, from the White Paper "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1".

There are no direct impacts on the hypervisor domain. It is expected that the management systems will communicate on the policies and SLA requirements to ensure adequate placement of applications/VMs.

8 Service Models

8.1 General

In order to meet network service performance objectives (e.g. latency, reliability), or create value added services for global customer or enterprises, it may be desirable for a Service Provider to be able to run VNFs on NFV Infrastructure (including infrastructure elements common with cloud computing services) which is operated by a different Service Provider.

The ability to remotely deploy and run virtualised network functions on NFV Infrastructure provided by another Service Provider permits a Service Provider to more efficiently serve its global customers. The ability for a Service Provider to offer its NFV Infrastructure as a Service (e.g. to other Service Providers) enables an additional commercial service offer (in addition to a Service Providers existing catalogue of network services that may be supported by VNFs) to directly support, and accelerate, the deployment of NFV Infrastructure. The infrastructure may also be offered as a Service from one department to another within a single Service Provider.

The commercial and deployment aspects of those agreements are out of the present document and only the technical requirements to facilitate these models will be depicted.

8.2 Deployment models

[**Mod.1]:** The NFV framework shall permit free grouping of virtualised network functions by the network operator (e.g. based on similar characteristics, service based, etc.).

[**Mod.2**]**:** The virtualisation framework shall provide the necessary mechanisms to achieve the same level of service availability for fully and partially virtualised scenarios as for existing non-virtualised networks.

[**Mod.3**]**:** Services (i.e. user facing services) making use of network functions shall be not be aware of actual implementation of it as virtual or non-virtual network function.

[**Mod.4]:** The virtualisation framework shall permit identification and management of the different types of traffic flows: services traffic over virtualised network functions from traffic running over non-virtualised functions, in addition to identifying traffic flows specific to resource facing operations (e.g. management, portability, scaling, etc.).

8.3 Service models

[**Mod.5**]**:** The NFV framework shall provide the appropriate mechanisms to facilitate network operators to consume NFV infrastructure resources operated by a different infrastructure provider, via standard APIs, and without degraded capabilities (e.g. scaling, performance, etc.) compared to consuming self -operated infrastructure resources.

[**Mod.6**]**:** The NFV framework shall permit Virtual Network Functions from different Service Providers to coexist within the same infrastructure while facilitating the appropriate isolation between the resources allocated to the different service providers.

[**Mod.7**]**:** The NFV framework should permit a Service Provider to fulfil, assure and bill for services delivered to end users across infrastructures that are independently administered.

[**Mod.8**]**:** The NFV framework shall be able to provide the necessary mechanisms to instantiate different virtualised network function components (e.g. VMs) of the same VNF on infrastructure resources managed by different administrative domains. MANO.

[**Mod.9**]**:** Appropriate mechanisms shall exist for:

- Authentication and authorization to control access in a way that only authorized VNFs from authorized Service Providers are allowed to be executed. Deferred to Security.
- Failure notification and diagnostics and measurement of SLA related parameters, so that VNF failures or resource demands from one Service Provider do not degrade the operation of other Service Provider's VNFs.
- Maintaining the relationships between each virtual network function/application, the service making use of it and the resources allocated to it. MANO and network.

[Mod.10]: The NFV framework shall support different options for allocation of resources to adapt to different sets of virtual network functions/applications, e.g. pay-as-you-go, overcommitting, immediate reservation, etc. mano.

Mod./H.1: In order to distribute or place VMs/Services there shall be communication between the VIM and the hypervisor.

Several operations are available in the communication between the VIM and the hypervisor. The following table is an illustrative list of the requests and responses between the VIM and the hypervisor to instantiate a VM.

Annex A of the present document contains multiple categories of communications between the VIM and the hypervisor to cover areas of the virtualised network from life cycle and configuration to monitoring, workload consolidation and live migration.

Table 9: Illustrative Virtual Machine Service Specification

Service Specification		Illustrative Parameters
Control Operations	Establishment	Request • server pool id • processing speed requirement • RAM requirement • number of vNIC with port speed/throughput requirements (including hypervisor I/O bypass eligibility) • storage requirements • resiliency requirements (e.g. fault prediction, memory mirroring) • affinity and anti-affinity rules (Multicore processor bitmask, referring to VNFC and VNF deployment) • migration eligibility • start and stop times, if applicable Return • instance id • vCPU and RAM characteristics • list of vNICs and characteristics • storage allocation • resiliency features • affinity and anti-affinity compliance
	Modification	Request • instance id • change to server pool (ie move the VM) • change to processing speed requirement • change to RAM requirement • change to number of vNIC with port speed/throughput requirements • change to storage requirements • change to resiliency requirements Return • instance id • vCPU and RAM characteristics • list of vNICs and characteristics • storage allocation • resiliency
	Removal	Request • instance id Return • success
Instance Interfaces	vNIC end point	VLANiD/MAC address
	Powering control	Power up/down/reboot
	Operational status	OAM parameters (running, stalled, crashed, etc.)
	Performance stats	OAM parameters (see dynamic metrics in clause 5.1.2, also VM activation and operation metrics in ETSI GS NFV INF 010 [i.16])
	Load VNFC Image	OVF file
Instance Functionality	Defined by VNFC Image	

Annex A (informative): Informative Reference: VIM Categories

This annex has information with regard to categories of the interface between the virtualisation infrastructure manager (VIM) and the hypervisor. It is not intended to be prescriptive, but informative as to the various operations and metrics provided by the hypervisor and VIM. It provides a starting point for the VIM and hypervisor communications as these communications are available today.

This contribution provides high level categories that are currently in use in virtualisation. Depending on how the user wishes to configure the systems/network will determine how the system gets utilized. For example, a lot of metrics are queried for, but in some cases the user may want to have the VIM get these data points automatically. This can be done thru policy settings. Alarms and events are generally sent automatically to the VIM from the hypervisor.

This contribution is focused on the VIM/Hypervisor communication. The contribution focus' on functions only and although contains numerous metrics, it is not all inclusive.

The VIM requires the following categories of information in order to have correct billing, security, performance, reliability to meet various SLAs effectively. Below is a figure depicting the major areas of communication between the hypervisor and the VIM, i.e. the southbound interface.

Table A.1: Definition/Description of the categories of VIM management towards INF

Category Name	Definition/Description
Logging for SLA, debugging	Metrics contributing to the performance, reliability information to provide operations/billing/charging information. This category also uses the metrics (and calculations) for debugging.
Host & VM configuration /Lifecycle	Tasks for starting, stopping configuration of servers, installing VMs, policy setting, termination of VMs, and placing objects in the VIM inventory/Database.
Database	As 'part' of the life cycle, objects are put in/out of the database.
Resources & VM inventory Management	Policies, alarms, events are associated with the objects that are in the database.
Events & Alarms	Data meant to initiate an action from the management. Events are records of user actions or system actions that occur on objects in a host. These events are supplied, not queried. Actions that might be recorded as events include, but are not limited to, the following examples: • A license key expires, a virtual machine is powered on, user logs in to a virtual machine, host connection is lost, and event data includes details about the event such as who generated it, when it occurred, and what type of event it is. There are generally three types of events: Informational, a warning or an error occurred. **Alarms** Alarms are notifications (not queried) that are activated in response to an event, a set of conditions, or the state of an inventory object. An alarm definition consists of the following elements: • Name and description - Provides an identifying label and description. • Alarm type - Defines the type of object that will be monitored. • Triggers - Defines the event, condition, or state that will trigger the alarm and defines the notification severity.
NETWORK/Connectivity	This category is a view into networking and the type of connections used by the INF.

Category Name	Definition/Description
Utilities	Distribution of Tasks, scheduling of workloads, consolidation of workloads, live migration, power management.
Memory & Storage	All metrics related to the memory/storage type, utilization, errors, faults.
CPUs, clusters, pools	All metrics related to the type, ids, frequencies, utilization of processors, cluster of processors or pools of processors, errors, faults.
User Access & controls	Secure Authentication and user controls.

A.1 Logging for SLA, debugging

The logging for SLA and debugging category is a category that has metrics contributing to the performance, reliability information to provide operations/billing/charging information. This category also uses the metrics (and calculations) for debugging.

These metrics/ information are gathered, collected & provided by various parameters, thus shown on top of the pile in the figure.

Some of the information required is contained in the following metrics with a specific section on alarms.

It is a gathering of several metrics across all the components in figure 1 that enable the calculations for the SLAs.

A.2 Host & VM configuration /Lifecycle for a VM from a VIM perspective

This category is centered on starting, configuration of servers, installing VMs, policy setting, termination of VMs, and placing objects in the VIM inventory/Database There are commands (sent to the INF), acknowledgements (sent from the INF) and results (sent from the INF) in order to implement the life cycle. This is considered to be first and last steps in many cases, thus the box is put closest to the INF. It does not mean the other categories are further away from the INF.

At a high level, currently the life cycle of a VM /system is started by the VIM communicating to the hardware thru the NF-Vi/C:

1) Thru the NF-Vi/C interface the VIM will configure the hardware, start the 'firmware, BIOS' and download the hypervisor.

2) Once installed the hypervisor, gets the processor, memory information, NIC drivers or other drivers, does configurations, does policy settings thru the VIM NF-Vi/H interface. And reports back to the VIM capabilities, system information. If there is not a communication the VIM knows there is a problem. If there is communication but any of the software has errors this to is reported.

3) Once there is a trusted state, the capabilities are known, the VIM can instantiate one or multiple VMs.

4) There is consistent communication between the VIM and the hypervisor. If the hypervisor stops communication the VIM takes action.

5) Based on policies, thresholds the hypervisor informs the VIM of regular and irregular activities. These activities or events can be from the hardware changing state, the bios being updated, or a fault condition has occurred.

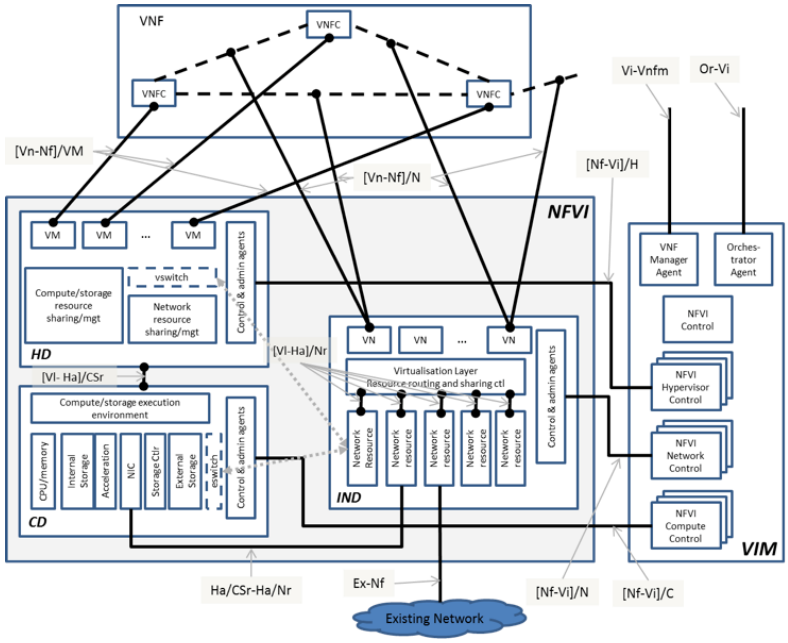

Figure A.1

Ultimately there are levels of Metrics, from high level KQIs: (how long does it take to start up the system, how long does it take to delete a system, etc.); all the way down to the hardware or compute domain. Or alternatively from the compute domain, there are hardware capabilities exposed into the software domain via registers, bios, OS, IPMI, drivers, up thru the virtualisation domain, which runs algorithms sent into the VIM for further calculations and to the Orchestrator for additional calculations to get to the KQIs.

Information models are used along the way to gather and communicate these metrics, results and performance.

Guest OS Performance

- some virtualisation processes dynamically allocate available resources depending on the status; or
- utilization rates, of virtual machines in the environment. This can make obtaining accurate information;
- about the resource utilization (CPU utilization, in particular) of individual virtual machines, or applications running within virtual machines, difficult. Hypervisors and VIMs provide virtual machine-specific performance counter libraries for the utilities that enable application administrators to access accurate virtual machine resource utilization statistics from within the utility. (such as Windows Perfmon).

These utilities take advantage of virtualised CPU performance counters to use performance tuning tools inside the guest operating system.

Host Health Stats

The host health monitoring tool allows a monitoring the health of a variety of host hardware components including:

- CPU processors, Memory, Fans, Temperature, Voltage, Power, Network, Battery, Storage, Cable/Interconnect, Software components, Watchdog, PCI devices.

The host health monitoring tool presents data gathered using something like Systems Management Architecture for Server Hardware (SMASH) profiles. The information displayed depends on the sensors available on the server hardware. SMASH is an industry standard specification providing protocols for managing a variety of systems. For more information, see http://www.dmtf.org/standards/smash.

A.3 Resources & VM inventory Management

Inventory Management is included in virtualisation today. As 'part' of the life cycle, components are put in/out of the database. Policies and practices are defined for the components.

In today's virtualisation environments, counters are used to ***query*** for statistics. A counter is a unit of information to a given inventory object or device. Statistics for each counter are rolled up after a specified collection interval. Each counter has several characteristics or attributes that are used to determine the statistical value collected. Collection levels determine the number of counters for which data is gathered during each collection interval. Collection intervals determine the time period during which statistics are aggregated, calculated, rolled up, and archived in the VIM database. Together, the collection interval and collection level determine how much statistical data is collected and stored in the VIM database.

A.4 Events, Alarms & Automated Actions (Data Provided by the INF to the VIM)

This category is in general data sent from the hypervisor to the VIM. Data meant to initiate an action from the management.

Events are records of user actions or system actions that occur on objects in a host. These events are supplied, not queried.

Actions that might be recorded as events include, but are not limited to, the following examples.

A license key expires, a virtual machine is powered on, user logs in to a virtual machine, host connection is lost, and event data includes details about the event such as who generated it, when it occurred, and what type of event it is.

There are generally three types of events: Informational, a warning or an error occurred.

Alarms

Alarms are notifications (not queried) that are activated in response to an event, a set of conditions, or the state of an inventory object. An alarm definition consists of the following elements:

- Name and description - Provides an identifying label and description.
- Alarm type - Defines the type of object that will be monitored.
- Triggers - Defines the event, condition, or state that will trigger the alarm and defines the notification severity.
- Tolerance thresholds - Provides additional restrictions on condition and state triggers thresholds that may be exceeded before the alarm is triggered.
- Actions - Defines operations that occur in response to triggered alarms. Predefined actions are specific to inventory object types.

Alarm definitions are associated with the object selected in the inventory. An alarm monitors the type of inventory objects specified in its definition. For example, the VIM could monitor the CPU usage of all virtual machines in a specific host cluster. The VIM can select the cluster in the inventory, and add a virtual machine alarm to it. When enabled, that alarm will monitor all virtual machines running in the cluster and will trigger when any one of them meets the criteria defined in the alarm. If necessary to monitor a specific virtual machine in the cluster, but not others, the VIM would select that virtual machine in the inventory and add an alarm to it. One easy way to apply the same alarms to a group of objects is to place those objects in a folder and define the alarm on the folder.

The rest of the document is design to share metrics that are gathered, or set by the VIM or provided by the INF automatically.

Parameters Used Today (Not All Inclusive, But Gives A Sense) For Operations (CPU, Memory, Guest, Hypervisor, Networking, Tasks).

Table A.2: General Metrics that are Queried/or set from the VIM to the NFVi/hypervisor

Description	functions
CPU	cpu entitlement, total mhz, usage (average), usage mhz, idle, reserved capacity
Disk	capacity, max Total Latency, provisioned, unshared, usage (average), used, the number of reads, the number of writes
Memory	consumed, memory entitlement, overhead, swap in Rate, swap out Rate, swap used, total mb, usage(average), vm mem ctl (balloon)
Network	usage (average), IPv6
System	heartbeat, uptime
Virtual Machine Operations	Number of changed host, storage

Storage Resources

Reports (queried or automatic from hypervisor) Reports display relationship tables that provide how an inventory object is associated with storage entities. They also offer summarized storage usage data for the object's virtual and physical storage resources. Reports are used to analyze storage space utilization, availability, multipathing status, and other storage properties of the selected object and items related to it.

Use of arrays that support storage vendor providers developed through Storage APIs/Storage Awareness, a reports view offers additional information about storage arrays, storage processors, ports, LUNs or file systems, and so on.

Storage topology maps visually represent relationships between the selected object and its associated virtual and physical storage entities.

A.5 Utilities (Data is either provided by INF hypervisor or queried from the VIM via the Nf-Vi)

Utilites provide methods to live migrate, scale, distribute or consolidate workloads.

A.6 CPU, Memory Data

Table A.3: Data supplied/queried (in general queried from the VIM)

Description	Functions
Resource pool ID or virtual machine ID of the resource pool or virtual machine that is running	Resource pool ID or virtual machine ID of the resource pool or virtual machine that is running (queried)
Name of the resource pool or virtual machine that is running	Name of the resource pool or virtual machine that is running
Number of members in the resource pool or virtual machine that is running	Number of members in the resource pool or virtual machine that is running
% utilized of physical CPU core cycles	Core cycles used by the resource pool, virtual machine, or cluster. % utilized might depend on the frequency with which the CPU core is running. When running with lower CPU core frequency, % utilized can be smaller than % run. On CPUs which support turbo mode, CPU frequency can also be higher than the nominal (rated) frequency, and % utilized can be larger than % run
Percentage of time spent in the Hypervisor Microkernel (microkernel runs on bare metal on the host, responsible for allocating memory, scheduling cpus and os services, etc.)	Time spent on behalf of the resource pool, virtual machine, or cluster to process interrupts and to perform other system activities**.** This time is part of the time used to calculate % utilized
% wait is percentage of time the resource pool, virtual machine, or cluster spent in the blocked or busy wait state	**Could be part of the idle time calculation**
% blocked	The total percentage of time the Resource Pool/Cluster spent in a blocked state waiting for events
% idle	Percentage of time the resource pool, virtual machine, or cluster was idle

Description	Functions
% Ready	Percentage of time the resource pool, virtual machine, or cluster was ready to run, but was not provided CPU resources on which to execute
% time the kernel did not run the VNF	Percentage of time the Hypervisor Microkernel deliberately did not run the resource pool, virtual machine, or cluster because doing so would violate the resource pool, virtual machine, or cluster's limit setting
% wait for swap of memory	Percentage of time a resource pool or cluster spends waiting for the Hypervisor Microkernel to swap memory
Event counts/s	Set of CPU statistics made up of per second event rates
Affinity	Bit mask showing the current scheduling affinity for the cluster, resource pool, cpu
Quarantined	Indicates whether cpu, cluster, resource pool is currently quarantined or not
Timer/s	Timer rate for cpu, cluster, resource pool
% sys time for scheduling	Percentage of system time spent during scheduling of a resource pool, virtual machine, or clusteron behalf of a different resource pool, virtual machine, or cluster while the resource pool, virtual machine, or cluster was scheduled
% deschedule	Percentage of time a resource pool spends in a ready, co-deschedule state
power	Current CPU power consumption for a resource pool (Watts)
% cpu resource contention	Percentage of time the resource pool or cluster was ready to run but was not scheduled to run because of CPU resource contention
% memory resource contention	Percentage of time the resource pool or cluster was ready to run but was not scheduled to run because of memory resource contention
Hypervisor page-sharing (Mbytes) **shared**	Amount of physical memory that is being shared
Hypervisor page-sharing (Mbytes) **common**	Amount of machine memory that is common across clusters
Hypervisor page-sharing (Mbytes) **saving**	Amount of machine memory that is saved because of page sharing
Current swap usage (Mbytes)	Rate at which memory is swapped in by the Hypervisor system from disk
Current compression usage (Mbytes)	Total compressed physical memory
Saved memory by compression(Mbytes)	Saved memory by compression
Total physical memory reclaimed	Total amount of physical memory reclaimed
Total amount of physical memory	Total amount of physical memory the hypervisor
Max amount of physical memory available for reclaim	Maximum amount of physical memory the Hypervisor host can reclaim
Memory reservation	Memory reservation for cluster, resource pool or virtual machine
Memory Limit	Memory limits for this cluster, resource pool or virtual machine
Memory	Memory shares for this cluster, resource pool or virtual machine
Current home node	The home node for the resource pool, cluster or vm
remote memory	Current amount of remote memory allocated to the virtual machine or resource pool. This statistic is applicable only on NUMA systems
Local memory	Current percentage of memory allocated to the virtual machine or resource pool that is local
Memory allocation	Amount of physical memory allocated to a resource pool or virtual machine
Guest physical memory	Amount of guest physical memory mapped to a resource pool or virtual machine
Machine memory allocation	Amount of machine memory the Hypervisor Microkernel wants to allocate to a resource pool or virtual machine
Memory working set	Working set estimate for the resource pool or virtual machine
Guest physical memory referenced	Percentage of guest physical memory that is being referenced by the guest. This can be an instantaneous value, an estimate, or a fast/slow moving average
Memory ballooning installed	Looks for a ballooning driver or limit
physical memory reclaimed via ballooning	Amount of physical memory reclaimed from the resource pool by way of ballooning
Hypervisor reclaiming memory via ballooning	Amount of physical memory the Hypervisor system attempts to reclaim from the resource pool or virtual machine by way of ballooning
Max memory reclaimable via ballooning thru the hypervisor	Maximum amount of physical memory the Hypervisor system can reclaim from the resource pool or virtual machine by way of ballooning. This maximum depends on the guest operating system type
Current Swap usage	Current swap usage by this resource pool or virtual machine. Target where the Hypervisor host expects the swap usage by the resource pool or virtual machine to be
Memory Swap rate	Rate at which the Hypervisor host swaps in memory from disk for the resource pool or virtual machine

Description	Functions
Rate at which memory is read from the host	Rate at which the Hypervisor host swaps resource pool or virtual machine memory to disk. The reads and writes are attributed to the hypervisor group only
Memory writes to the host cache	Rate at which memory is written to the host cache from various sources
Checkpoint file read data	Amount of data read from checkpoint file
Size of checkpoint file	Size of checkpoint file
Pages that are zeroed	Resource pool or virtual machine physical pages that are zeroed
Pages that are shared	Resource pool or virtual machine physical pages that are shared
Pages that are saved	Machine pages that are saved because of resource pool or virtual machine shared pages
Space overhead for pool	Current space overhead for resource pool
Max space limit	Maximum space overhead that might be incurred by resource pool or virtual machine
Space for cluster	Current space overhead for a user cluster

Interrupt Metrics

The interrupt metrics displays information about the use of interrupt vectors.

Interrupt vector ID.

Total number of interrupts per second. This value is cumulative of the count for every CPU.

Interrupts per second on CPU x.

Average processing time per interrupt (in microseconds).

Average processing time per interrupt on CPU x (in microseconds).

Devices that use the interrupt vector.

Storage Adapter Metrics.

Name of the storage adapter.

Storage path name.

Number of paths.

Current queue depth of the storage adapter.

Number of commands issued per second.

Number of read commands issued per second.

Number of write commands issued per second.

Megabytes read per second.

Megabytes written per second.

Number of SCSI reservations per second.

Number of SCSI reservation conflicts per second.

Average device latency per command, in milliseconds.

Average Hypervisor Microkernel latency per command, in milliseconds.

Average virtual machine operating system latency per command, in milliseconds.

Average queue latency per command, in milliseconds.

Average device read latency per read operation, in milliseconds.

Average Hypervisor Microkernel read latency per read operation, in milliseconds.

Average guest operating system read latency per read operation, in milliseconds.

Average queue latency per read operation, in milliseconds.

Average device write latency per write operation, in milliseconds.

Average Hypervisor Microkernel write latency per write operation, in milliseconds.

Average guest operating system write latency per write operation, in milliseconds.

Average queue latency per write operation, in milliseconds.

Number of failed commands issued per second.

Number of failed read commands issued per second.

Number of failed write commands issued per second.

Megabytes of failed read operations per second.

Megabytes of failed write operations per second.

Number of failed SCSI reservations per second.

Number of commands aborted per second.

Number of commands reset per second.

The number of PAE (Physical Address Extension) commands per second.

The number of PAE copies per second.

The number of split commands per second.

The number of split copies per second.

Allows view storage resource utilization statistics broken down by individual paths belonging to an storage adapter.

Storage Device Metrics.

By default, the information is grouped per storage device, including group the statistics per path, per cluster, or per partition.

Storage Device Statistics.

Name of the storage device.

Path name.

Cluster ID. This ID is visible only if the corresponding device is expanded to clusters.

The cluster statistics are per cluster per device.

Partition ID. This ID is visible only if the corresponding device is expanded to partitions.

Number of paths.

Number of clusters.

Number of partitions.

Number of shares. This statistic is applicable to clusters.

Block size in bytes.

Number of blocks of the device.

Current device queue depth of the storage device:

- Number of commands in the Hypervisor Microkernel that are currently active.
- Number of commands in the Hypervisor Microkernel that are currently queued.
- Percentage of the queue depth used by Hypervisor Microkernel active commands.
- LOAD Ratio of Hypervisor Microkernel active commands plus Hypervisor Microkernel queued commands to queue depth.
- Number of commands issued per second.
- Number of read commands issued per second.

Number of write commands issued per second.

Megabytes read per second.

Megabytes written per second.

Average device latency per command in milliseconds.

Average Hypervisor Microkernel latency per command in milliseconds.

Average guest operating system latency per command in milliseconds.

Average queue latency per command in milliseconds.

Average device read latency per read operation in milliseconds.

Average Hypervisor Microkernel read latency per read operation in milliseconds.

Average guest operating system read latency per read operation in milliseconds.

Average queue read latency per read operation in milliseconds.

Average device write latency per write operation in milliseconds.

Average Hypervisor Microkernel write latency per write operation in milliseconds.

Average guest operating system write latency per write operation in milliseconds.

Average queue write latency per write operation in milliseconds.

Number of commands aborted per second.

Number of commands reset per second.

Number of PAE commands per second. This statistic applies to only paths.

Number of PAE copies per second. This statistic applies to only paths.

Number of split commands per second. This statistic applies to only paths.

Number of split copies per second. This statistic applies to only paths.

Storage

Allows view of storage resource; utilization statistics separated by individual devices belonging to an expanded storage device.

The statistics are per geo per device.

Expand or roll up storage path statistics. Supplied storage resource utilization statistics separated by individual paths belonging to an expanded storage device.

Expand or roll up storage partition statistics. This command allows view storage resource.

Utilization statistics separated by individual partitions belonging to an expanded storage device.

Virtual Machine Storage

Virtual machine-centric storage statistics.

By default, statistics are aggregated on a per-resource-pool basis. One virtual machine has one.

corresponding resource pool, so the panel displays statistics on a per-virtual-machine or cluster basis.

Description of metrics

Resource pool ID or VSCSI ID of VSCSI device.

Resource pool ID.

Name of the resource pool.

Name of the VSCSI device.

Number of VSCSI devices.

Number of commands issued per second.

Number of read commands issued per second.

Number of write commands issued per second.

Megabytes read per second.

Megabytes written per second.

Average latency (in milliseconds) per read.

Average latency (in milliseconds) per write.

A.7 NETWORK/Connectivity

This category is a view into networking and the type of connections used by the INF.

Metrics

Statistics are arranged by port for each virtual network device configured.

Virtual network device port ID.

Corresponding port is an uplink.

The corresponding link is up.

Link speed in Megabits per second.

Corresponding link is operating at full duplex.

Virtual network device port user.

Virtual network device type. (hub or switch).

Virtual network device name.

Number of packets transmitted per second.

Number of packets received per second.

MegaBits transmitted per second.

MegaBits received per second.

Percentage of transmit packets dropped.

Percentage of receive packets dropped.

Name of the physical NIC used for the team uplink.

Number of multicast packets transmitted per second.

Number of multicast packets received per second.

Number of broadcast packets transmitted per second.

Number of broadcast packets received per second.

Description of relationships between physical entities and logical entities managed by the same MIB agent. See IETF RFC 4133 [i.2] for more information.

MIB Defines storage, device, and filesystem types for use with IEEE 802.11 [i.7].

MIB Defines objects for managing devices that support IEEE 802.1D [i.3].

IEEE 802.3 - MIB [i.9] Defines objects for managing devices that support IEEE 802.3ad [i.8] link aggregation.

IEEE8021-Q -MIB [i.4] Defines objects for managing Virtual Bridged Local Area.

Networks

Mib defines attributes related to physical NICs on the host system.

Mib defines objects for managing implementations of the Internet Protocol (IP) in an IP version independent manner.

Mib defines objects for managing IP forwarding.

Mib Defines objects for managing devices using Linked Layer Discovery Protocol (LLDP).

Mib Defines conformance groups for MIBs.

Mib Defines textual conventions.

Mib Defines objects for managing devices using the TCP protocol.

Mib Defines objects for managing devices using the UDP protocol.

Annex B (informative): Informative reference

The below Contribution needs updating and inaccuracies fixed. It is left in the present document for history purposes, only.

Features of the Domain Affecting Performance.

Data plane performance with SR-IOV and Live Migration.

This is a reformatted version of contribution NFVINF(13)VM_019_Data_plane_performance from Metaswitch Networks.

Statement of the issue

A key goal of the NFV is to permit network functions with high data plane performance requirements to be implemented as VNFs - the belief that that was technically possible was cited as one of the founding justifications for the formation of the NFV, and without it the rationale for the whole NFV enterprise is greatly diminished.

As part of that, the NFV INF Compute & Hypervisor domains currently call for NFV support for the current state-of-the-art mechanisms to permit this, including SR-IOV and DPDK.

However, they also deem as essential support for cloud features such as live migration which are important for HA.

Opinion Technical background

Current state of play

The local network stack consists of a VM connecting through a virtual NIC driver to a software vSwitch running in the hypervisor, with the hypervisor having exclusive control of the physical NIC. Packets flow through the hypervisor and vSwitch, resulting in additional packet handling and copying which results in very poor data plane performance from within the VM. In our experience with virtualising session border control function, this architecture typically results in large performance degradation compared to bare metal performance, on occasion order-of-magnitude.

The current state-of-the-art in achieving higher virtualised performance is to use SR-IOV or direct i/o from the VM to the NIC, bypassing the hypervisor and vSwitch. This requires support on the NIC to effectively present itself as multiple virtual NICs) which is common but not universal.

By bypassing the hypervisor and the vSwitch, and in conjunction with techniques such as DPDK which improve the efficiency of packet passing in both the virtualised and non-virtualised case, near bare metal data plane performance can be achieved from within a VM.

However, this comes at a cost: precisely because the packet path now bypasses the hypervisor and vSwitch, the hypervisor has lost the indirection layer it requires to implement features such as seamless and transparent VM migration between hosts. For example, in a VMware environment vMotion, High Availability, Suspend and resume and many other functions may be lost (see http://pubs.vmware.com/vsphere-51/index.jsp?topic=%2Fcom.vmware.vsphere.networking.doc%2FGUID-E8E8D7B2-FE67-4B4F-921F-C3D6D7223869.html).

Further, while all the common hypervisors (ESXi, Xen, KVM) support SR-IOV, support for configuring it in higher-level cloud stacks is sparse - possibly, because of the limitations it may imposes. It is interesting to note that there has been a proposal and prototype code for SR-IOV support circulating in the community for ~2 years but it is still not on the roadmap for mainline.

There are some mooted techniques for working around these limitations: for instance, notifying a VM of an impending live migration, requiring it to detach its vNIC device in response, moving the VM between hosts, then notifying it again to allow it to reattach. However, none of these techniques meet the "seamless and transparent" requirement at this time.

B.1 Future direction

There appear to be two different approaches for addressing the issues identified in the present document, namely improving the performance of the path between the physical network and the VM through the vSwitch, and addressing the limitations of SR-IOV in respect of live migration.

B.1.1 Improving vSwitch Performance

Currently, there is a considerable amount of industry activity focusing on the performance limitations of vSwitches. This is occurring at a minimum in two areas.

Software vSwitch developers are trying to improve the performance of the NIC -> vSwitch path by exploiting techniques such as DPDK (e.g. Intel have demonstrated integration of DPDK with Open vSwitch).

Hardware vendors are looking to add vSwitch function, whether on NICs, in innovative server form factors combining x86 & packet processors, or by pushing vSwitch function down to the external physical switch. These typically expose northbound OpenFlow interfaces for control.

It is, however, unclear to what extent these approaches can also improve the performance of the vSwitch -> VM path which would be necessary to achieve acceptable data plane performance. Clearly, any techniques would only be of interest to NFV if they were standard rather than proprietary.

Furthermore, it seems unlikely that software-based techniques for improving the performance of the vSwitch -> VM path will be able to approach the levels of performance seen with SR-IOV. And techniques that require the development of new hardware or the deployment of specialized packet processing silicon inside servers would not appear to be compatible with NFV's objectives to leverage industry-standard generic server hardware.

B.1.2 Addressing Limitations of SR-IOV

The existence of a direct hardware path between a VM and a physical NIC that is a key characteristic of SR-IOV means that the hypervisor is not in position to support live migration of VMs in a manner that is transparent to the VM and its guest application.

However, that does not imply that it is impossible to support live migration with SR-IOV. It means that live migration with SR-IOV can only be achieved with the assistance of software running in the VM. It is suggested that a solution might be possible with the following ingredients.

Enhancements to the SR-IOV vNIC network drivers in the guest OS intended to support live migration under the control of the cloud stack. These network drivers would need to support APIs enabling external control by the cloud stack.

Enhancements to the cloud stack to control vNIC network drivers in the guest OS.

These techniques have in fact been put into practice. At least one virtualisation environment supports live migration with SR-IOV, namely Microsoft Hyper-V. See http://msdn.microsoft.com/en-gb/library/windows/hardware/hh440249(v=vs.85).aspx.

There have also been papers about and demonstrations of live migration with SR-IOV. Intel published a research paper entitled "Live Migration with Pass-through Device for Linux VM" in 2008. This describes an approach applicable to KVM and Xen. And VMware showed a demonstration of vMotion with SR-IOV in 2009, as described in this article http://www.vcritical.com/2013/01/sr-iov-and-vmware-vmotion/ which also points out that VMware have so far not chosen to incorporate this capability into their product.

It should perhaps be pointed out that all of the techniques described for supporting live migration with SR-IOV do introduce short term reductions in data plane performance during the migration process. Although such live migration could be described as "transparent" at least as far as the guest application is concerned, it could not be described as totally seamless. It will be for the NFV to evaluate the trade-offs involved.

Implications for NFV requirements

As described above, the main implication for NFV is that some of the requirements currently set out by the INF WG are inconsistent with one another because, in general, it is not currently possible to support live migration with SR-IOV.

However, depending upon the resolution of this issue there may be wider implications. For example, RELAV is considering the issue of how the VNFI infrastructure is upgraded. From NFV Availability Straw man v2:

Software Management

Upgrade of individual vNFs can be handled internally by the vNF. The requirements on the services within the infrastructure need to be set. Should they all have the requirement to do a soft upgrade without disturbances for the users of the service or is it ok to move applications away and then upgrade the service?

The approach to move applications away and then upgrade would not be feasible (at least, not without service interruption) if some of the hosted VNFs used the SR-IOV approach, at least in the absence of the enhancements described above. More generally, the potential loss of live migration and other features may have an impact on a range of approaches being considered in MANO and RELAV which may assume they are all available.

It would appear that the only way to reconcile these apparently conflicting requirements is to anticipate the development by the industry of solutions that support SR-IOV together with live migration.

Suggested Approach

Multiple WGs within the NFV initiative appear to have recognized that SR-IOV is currently an essential element of any solution that supports VNFs with an acceptable level of data plane performance. However, support for live migration is also an essential requirement, and today these two requirements cannot be met at the same time - except in the case of one virtualisation environment, namely Microsoft Hyper-V, and even then not seamlessly.

The INF WG sets out specific requirements to encourage the industry to support live migration in conjunction with SR-IOV, with the aim of minimizing, if not eliminating, temporary degradation of data plane performance during the migration process.

The INF WG sets out specific requirements to encourage the industry to improve the performance of the data path between the network and the VM via software-based vSwitches, as an alternative approach to SR-IOV. An optimized vSwitch approach may be preferred in cases where temporary degradation of data plane performance during migration is not acceptable, or in cases where the cloud stack is not capable of managing live migration in conjunction with SR-IOV.

It may take the industry some time to deliver solutions that support SR-IOV and live migration, and until that time it may be necessary to accommodate two classes of VNFs, one that is intended to work with SR-IOV and one that is not - but which can support live migration under the control of the cloud stack.

The INF WG, specifically the Compute and Hypervisor domains, defines two classes of VNF - one using SR-IOV (say "pass-through VNF"), one not - and defines the functional limitations imposed by the first class.

The INF WG works with MANO and RELAV and other relevant WGs to define the impact of pass-through VNFs on their work e.g. different requirements on what the NFV Orchestrator can do for the two classes of VNF, different overall availability requirements.

Both VNFs and VNFIs declare which class(es) they support (e.g. in the Resource Requirement Description and Resource Availability Description of RELAV).

If a VNF supports both classes, it could clearly state the performance difference between the two so that providers can fully weigh the costs and benefits of the two approaches - it is not clear that it will always be in the provider's interest to deploy a data plane VNF using the SR-IOV approach, even though it uses less host hardware resources.

The INF WG (Hypervisor & Network Domains) set out requirements for a common high-performance interface from VMs to a vSwitch that is independent of the implementation of the vSwitch as software in the hypervisor or hardware on the NIC or elsewhere.

Annex C (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Valerie Young, Intel Corporation

Other contributors:

Bhavesh Davda, Vmware
Michael Bursell, Citrix
Andy Reid, BT
Steven Wright, ATT
Al Morton, ATT
Yun Chao, Huawei
Trevor Cooper, Intel
Richard Phelan, ARM
Donna Reineck, Vmware
Mike Young, Huawei
Michael Rooke, NSN
Matthias, Ericsson
Jan Ellsberger, Ericsson
Larry Pearson, ATT
Bruno Chatras, Orange

History

Document history		
V1.1.1	January 2015	Publication