ETSI GS NFV-SWA 001 V1.1.1 (2014-12)

Network Functions Virtualisation (NFV); Virtual Network Functions Architecture

Reference
DGS/NFV-SWA001

Keywords
architecture, functional, NFV, requirements

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

Foreword

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

Modal verbs terminology

1 Scope

The present document objective is to identify the most common and relevant software architectural patterns present when virtualising network functions and therefore to identify and specify functional requirements necessary to enable such patterns. The information consolidated is reflecting the experience from vendors and operators going through virtualisation of a number of network functions, with a focus on the use case list provided by the NFV Use Cases GS document [i.7].

The present document describes the Network Function Virtualisation abstract software architecture comprising of the following topics:

- Defining the functions, and interfaces of software architecture relative to the NFV overall architecture.
- Supporting Management and Orchestration Functional requirements.
- Supporting Infrastructure requirements.
- Describing best practices for NFV Design.
- Functional Decomposition types and use cases.

The present document does not provide any detailed specification. However, the present document makes reference to specifications developed by other bodies, gap, and to potential specifications.

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.

Not applicable.

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

[i.2] ETSI GS NFV-INF 005: "Network Functions Virtualisation (NFV); Infrastructure; Network Domain".

[i.3] ETSI GS NFV-MAN 001: "Network Functions Virtualisation (NFV); Management and Orchestration".

[i.4] ETSI GS NFV 002 (V1.1.1): "Network Functions Virtualisation (NFV); Architectural Framework".

[i.5] ETSI GS NFV 004 (V1.1.1): "Network Functions Virtualisation (NFV); Virtualisation Requirements".

[i.6] ETSI GS NFV 003 (V1.1.1): "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.7] ETSI GS NFV 001 (V1.1.1): "Network Functions Virtualisation (NFV); Use Cases".

[i.8] ETSI GS NFV-PER 001: "Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practises".

[i.9] ETSI GS NFV-REL 001: "Network Functions Virtualisation (NFV); Resiliency Requirements".

[i.10] Open Data Center Alliance, ODCA Service Orchestration Master Usage Model, ODCA.

NOTE: Available at http://www.opendatacenteralliance.org/docs/ODCA_Service_Orch_MasterUM_v1.0_Nov2012.pdf.

[i.11] ETSI TS 123 228: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; IP Multimedia Subsystem (IMS); Stage 2 (3GPP TS 23.228)".

[i.12] ETSI TS 123 218: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; IP Multimedia (IM) session handling; IM call model; Stage 2 (3GPP TS 23.218)".

[i.13] ISO/IEC 42010:2011: "Systems and Software Engineering - Architecture Description".

[i.14] ATIS-I-0000044 (October 2013): "Operational opportunities and challenges of SDN/NFV programmable infrastructure", section 4.2.1.1 "Service Provider Devops".

[i.15] ETSI TS 123 203: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control architecture (3GPP TS 23.203)".

[i.16] ETSI TS 132 251: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Telecommunication management; Charging management; Packet Switched (PS) domain charging (3GPP TS 32.251)".

[i.17] ETSI TS 132 240: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Telecommunication management; Charging management; Charging architecture and principles (3GPP TS 32.240)".

[i.18] Open Network Foundation, ONF.

NOTE: Available at https://www.opennetworking.org/sdn-resources/sdn-definition.

[i.19] PCI Single Route I/O Virtualization (SR-IOV). [Online].

NOTE: Available at http://www.pcisig.com/specifications/iov/.

[i.20] Recommendation ITU-T H.248.1 (03/2013): "Gateway control protocol: Version 3".

[i.21] Technical Report Draft TR H.Sup.OpenFlow (2014): "Protocol evaluation - OpenFlow versus H.248".

NOTE: Latest draft available at http://wftp3.itu.int/av-arch/avc-site/2013-2016/1403_Gen/TD-19.zip.

[i.22] Recommendation ITU-T M.3050.1 (03/2007): "Enhanced Telecom Operations Map (eTOM) - The business process framework".

[i.23] Recommendation ITU-T M.3010 (02/2000): "Principles for a telecommunications management network".

[i.24] Recommendation ITU-T M.3400 (02/2000): "TMN management functions".

[i.25] Recommendation ITU-T X.700 (09/1992): "Management framework for Open Systems Interconnection (OSI) for CCITT applications".

[i.26] ETSI GS NFV-PER 002 (V1.1.1): "Network Functions Virtualisation (NFV); Proofs of Concepts; Framework".

[i.27] IETF RFC 3031 (January 2001): "Multiprotocol Label Switching Architecture", E. Rosen, A. Viswanathan and R. Callon.

[i.28] IETF RFC 3069 (February 2001): "VLAN Aggregation for Efficient IP Address Allocation", D. McPherson and B. Dykes.

[i.29] IETF RFC 3809 (June 2004): "Generic Requirements for Provider Provisioned Virtual Private Networks (PPVPN)", A. Nagarajan.

[i.30] IETF RFC 4385 (February 2006): "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", S. Bryant, G. Swallow, L. Martini and D. McPherson.

[i.31] IETF RFC 4464 (May 2006): "Signaling Compression (SigComp) Users' Guide", A. Surtees and M. West.

[i.32] IETF RFC 4761 (January 2007): "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", K. Kompella and Y. Rekhter.

[i.33] Recommendation ITU-T Y.3300 (06/2014): "Framework of software-defined networking".

[i.34] ETSI TS 129 333: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Multimedia Resource Function Controller (MRFC) - Multimedia Resource Function Processor (MRFP) Mp interface; Stage 3 (3GPP TS 29.333)".

3 Definitions and abbreviations

3.1 Definitions

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

compute node: See ETSI GS NFV-INF 001 [i.1].

Network Function (NF): See ETSI GS NFV 003 [i.6].

Network Function Virtualisation Infrastructure (NFVI): See ETSI GS NFV 003 [i.6].

Network Functions Virtualisation Orchestrator (NFVO): See ETSI GS NFV-MAN 001 [i.3].

NF Forwarding Graph: See ETSI GS NFV 003 [i.6].

NF Set: See ETSI GS NFV 003 [i.6].

Physical Network Function (PNF): See ETSI GS NFV 003 [i.6].

network service: See ETSI GS NFV 003 [i.6].

Virtual Machine (VM): See ETSI GS NFV 001 [i.1].

virtualisation container: partition of a compute node that provides an isolated virtualised computation environment

NOTE: Examples of virtualisation container includes virtual machine and OS container.

Virtualisation Deployment Unit (VDU): See ETSI GS NFV-MAN 001 [i.3].

Virtualised Network Function (VNF): See ETSI GS NFV 003 [i.6].

Virtualised Network Function Component (VNFC): See ETSI GS NFV 003 [i.6].

Virtualised Network Function Component (VNFC) Instance: See ETSI GS NFV 003 [i.6].

VNF Descriptor (VNFD): See ETSI GS NFV-MAN 001 [i.3].

VNF Forwarding Graph (VNF-FG): See ETSI GS NFV 003 [i.6].

VNF Instance: See ETSI GS NFV-MAN 001 [i.3].

VNF Network Connectivity Topology (VNF-NCT): graph that defines the connectivity topology among (v)NFs by describing how its nodes are connected to one another

VNF Package: See ETSI GS NFV-MAN 001 [i.3].

VNF Provider: entity that provides VNF Package(s)

VNF Set: See ETSI GS NFV 003 [i.6].

3.2 Abbreviations

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

ADC	Application Detection and Control
API	Application Programming Interface
AppVM	Application Virtual Machines
ATM	Asynchronous Transfer Mode
BFCP	Binary Floor Control Protocol
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol - Link State
BRAS	Broadband Remote Access Server
CDN	Content Delivery Network
COTS	Commercial off the Shelf
CPU	Central Processing Unit
CSCF	Call Session Control Function
DB	DataBase
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Nane Server
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DRA	Diameter Routing Agent
DSCP	Differentiated Services Code Point
DSP	Digital Signal Processor
DSR	Direct Server Return
ELAN	Ethernet Virtual Private LAN
EM	Element Management
EPC	Evolved Packet Core
ETH	Ethernet
eTOM	enhanced Telecom Operations Map
EVPN	Ethernet VPN
FAB	Fulfilment, Assurance, Billing
FCAPS	Fault, Configuration, Accounting, Performance, Security
FORCES	FOrwarding and Control Element Separation
FRR	Fast ReRoute
GGSN	Gateway GPRS Service Node
GTP	GPRS Tunnel Protocol
GW	Gateway
HDW	Hardware
HTTP	Hypertext Transfer Portocol
HW	Hardware

I2SR	Interface 2 the Routing System
IB	Infiniband
IDPS	Intrusion Detection And Prevention Systems
IGMP	Internet Group Management Protocol
IGP	Internet Gateway Protocol
IMS	IP Multimedia Subsystem
IO	Input Output
IP	Internet Protocol
IPS	Intrusion Prevention System
ISA	Industry Standard Architecture
ISO	International Organisation for Standardization
IT	Information Technology
LAG	Link Aggregation Groups
LAN	Local Area Network
LB	Load Balancer
LISP	Location Identifier Separation Protocol
LSP	Label Switched Paths
MAC	Media Access Control
MEF	Metro Ethernet Forum
MME	Mobility Management Entity
MPLS	Multiprotocol Label Switching
MRB	Media Resource Broker
MRF	Media Resource Function
MRF-C	Multimedia Resource Function Controller
MRF-P	Multimedia Resource Function Processor
MSRP	Message Session Relay Protocol
NAT	Network Address Translation
NF	Network Function
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NFVO	Network Functions Virtualisation Orchestrator
NIC	Network Interface Controller
NVFI	Network Functions Virtualisation Infrastructure
NVGRE	Network Virtualisation using Generic Routing Encapsulation
OAM	Operations, Administration and Maintenance/Management
OCS	Online Charging Function
OF	OpenFlow
OFCS	Offline Charging Function
OFLS	OpenFlow Logical Switch
ONF	Open Networking Foundation
OS	Operating System
OSS	Operations Support System
PCC	Policy and Charging Control
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
PGW	Packet Data Network Gateway
PMIP	Proxy Mobile IP
PNF	Physical Network Function
RAM	Random Access Memory
RDBMS	Relational Data Base Management System
REQ	Requirement
RFC	Request For Comment
RLOC	Routing Locator
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SCSI	Small Computer System Interface
SDK	Software Development Kit
SDN	Software Defined Networks
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SNMP	Signalling Network Management Protocol
SSL	Secure Socket Layer

SWA	Software Architecture
TCP	Transfer Control Protocol
TDF	Traffic Detection Function
TDF-C	Traffic Detection Function - Control
TDF-LB	Traffic Detection Function - Load Balancer
TDF-P	Traffic Detection Function - Processing
TE	Traffic Engineering
TMN	Telecommunications Management Network
UDP	Unreliable Datagram Protocol
VDC	Virtual Data Centre
VIM	Virtualised Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtualised Network Function
VNFC	Virtualised Network Function Component
VNFD	Virtualised Network Function Descriptor
VNFFG	VNF Forwarding Graph
VNFM	Virtualised Network Function Manager
VNF-NCT	VNF Network Connectivity Topology
vNIC	Virtualised NIC
VoLTE	Voice over LTE
VPLS	Virtual Private LAN Service
VxLAN	Virtual eXtensible LAN
WAF	Web Application Firewall
WAN	Wide Access Network

4 Overview of VNF in the NFV Architecture

4.1 Introduction

A Virtualised Network Function (VNF) is a functional element of the NFV architecture framework [i.4] as represented on figure 1. Reference points in-scope of the present document are those between a VNF and a VNF Manager (Ve-Vnfm) and between a VNF and NFVI (Vn-Nf), see red dashed line circle in figure 1. The present document addresses functional requirements for virtualising network functions in the form of software components deployed within an NFVI, in support of the deployment of network services.

Software architectures describe the functionality of software systems from the viewpoints of various stakeholders [i.13]. ETSI NFV-PER 002 [i.26] identified Proof of Concept Framework stakeholders from the membership categories (e.g. Service Provider, Network Operator, and Manufacturer) defined in the ETSI Directives. A Manufacturer of VNFs may have particular concerns with the software development aspects in creation of VNFs. A Network Operator may have particular concerns with the efficient deployment and operation of VNFs within his NFVI. A Service Provider may have particular concerns with the fulfilment, assurance and billing of services based on VNFs delivered to end users. ETSI GS NFV 004 [i.5] also identifies requirements impacting functions based on the roles, as defined in [i.5] Requirement [Sec. 5] of the actors initiating operations. Such roles impacting the VNF may extend beyond those identified by the ETSI Directives' membership categories, as defined in [i.7]. Commercial entities may need to act in multiple roles in order to meet certain NFV objectives; for example the objectives of NFV for automation, as defined in [i.5], Requirement [OaM.1] and rapid service innovation and deployment, as defined in [i.4], clause 4.2 may lead some entities to consider "devops" [i.14] to automate the process of development and deployment of VNFs and the end-end network services constructed from them.

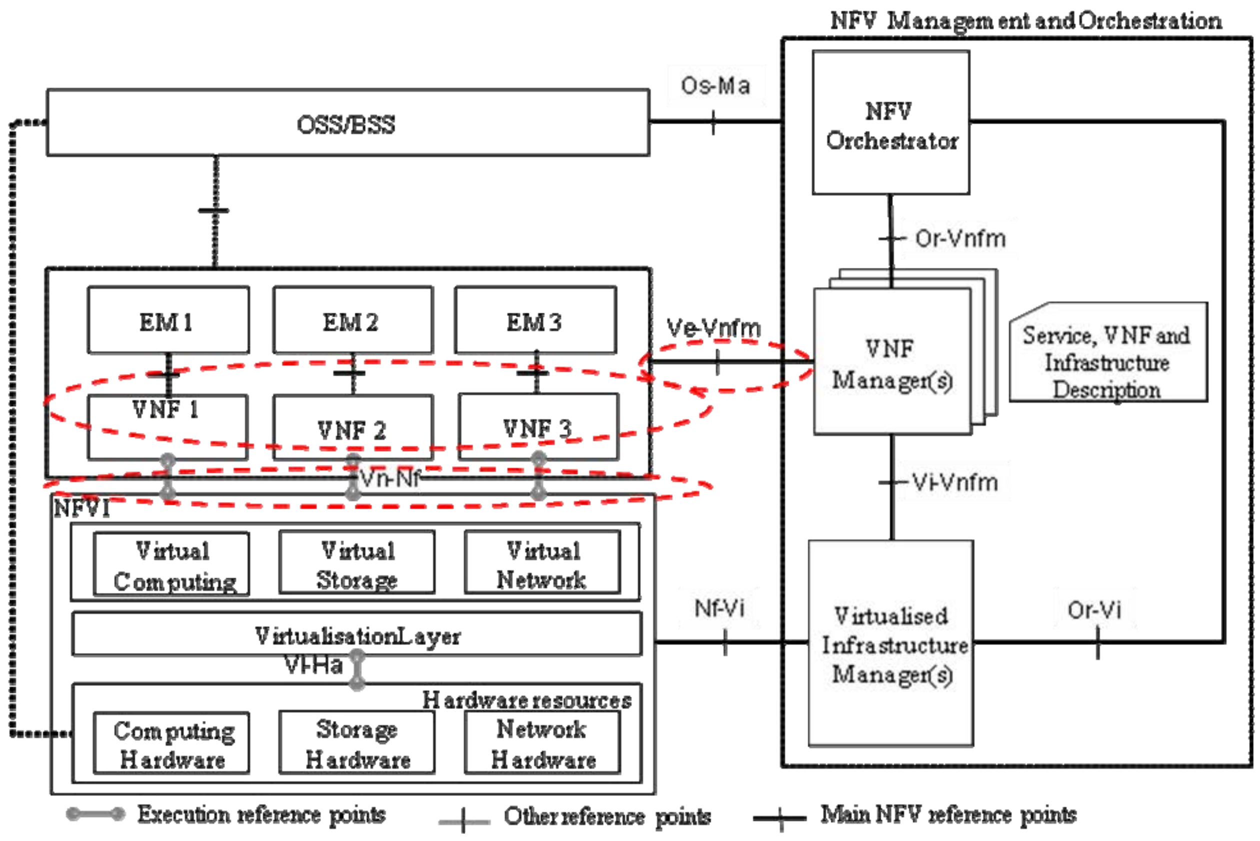

Figure 1: NFV Software Architecture Scope within the NFV Reference Architecture Framework

As defined in [i.6] a network service is a composition of network functions, in the form of a network functions sets and/or network forwarding graphs. The present document also addresses the requirements for deploying network services involving at least one virtual network function.

The present document supports distributed VNF deployment models in support of end-to-end network services as defined in [i.5], Requirement "Mod7", delivered across multiple NFVI Nodes deployed in disparate NFVI-PoPs as defined in [i.5], Requirement "Port. 1". It shall be possible to deploy end-to-end network services across independently operated NFVI nodes as defined in Use Case #1 (NFVIaaS) in [i.7] and Requirement "Mod 7" in [i.5], a mix of NFVI Nodes and non-virtualised PNFs as defined in [i.5] Requirements "Mod 2", "Mig.1", and "Gen.4" and coexist with other network services deployed in parallel in the same NFVI as defined in [i.5] Requirements "Mod. 10" and "Sec. 1".

4.2 VNF Architecture

Figure 2 shows the internal architecture of a VNF. It provides more details on the entities and interfaces relevant for the discussion in the present document while deliberately leaving out those aspects that fall into the infrastructure and management and orchestration domains. See [i.3] and [i.1] for details on those.

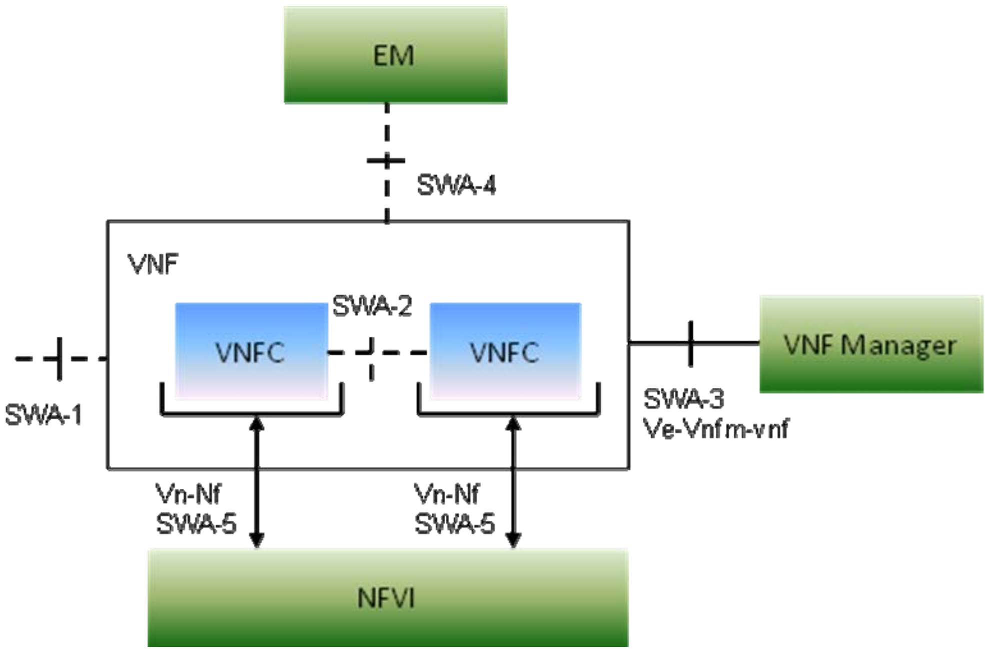

Figure 2: Functional View

A Virtualised Network Function (VNF) is a Network Function capable of running on an NFV Infrastructure (NFVI) and being orchestrated by a NFV Orchestrator (NFVO) and VNF Manager. It has well-defined interfaces to other NFs via SWA1, the VNF Manager, its EM, and the NFVI and a well-defined functional behaviour.

A VNF may implement a single network entity with interfaces and behaviour defined by standardisation organizations like e.g. 3GPP or IETF, while another VNF may implement groups of network entities. When a group of entities is implemented the internal interfaces between them do not need to be exposed. In both cases the VNFs will provide well-defined interfaces and behaviour.

EXAMPLE: A P-GW network entity may be composed with other network entities like MME and S-GW into an "EPC VNF". An "EPC VNF" is not restricted to staying compatible to 3GPP-defined interfaces between the internal entities, but will provide the well-defined external interfaces of the group. Likewise, VNFs may implement smaller functional units like TDF or PCEF.

When designing and developing the software that provides the VNF, VNF Providers may structure that software into software components (implementation view of software architecture) and package those components into one or more images (deployment view of software architecture). In the following, these VNF Provider defined software components are called VNF Components (VNFCs). VNFs are implemented with one or more VNFCs and it is assumed, without loss of generality, that VNFC Instance map 1:1 to NFVI Virtualised Container interface (Vn-Nf, see figure 2).

How a VNF Provider structures a VNF from a set of VNFCs depends on many factors, e.g. the prioritization of performance, scalability, reliability, security and other non-functional goals, the integration of components from other VNF Providers, operational considerations, the existing code base, etc. In general, no two VNF Providers will structure a VNF from the same VNFCs.

Another typical characteristic of VNFCs is that their functionality and interfaces between VNFCs change over time as part of the continuous innovation process. For example, a VNF Provider may decide to improve the VNF's performance by enhancing the interface between two VNFCs, but these changes do not have effects outside the VNF. As a consequence, VNFCs can in general not be assumed to have interfaces and functional behaviour that are meaningful outside of the VNF Provider's software development context. Their interfaces and functional behaviour are neither well-defined nor necessarily stable from one software release to another.

A VNF is an abstract entity that allows the software contract to be defined, and a VNF Instance is the runtime instantiation of the (design time) VNF. A VNFC is a VNF Providers specific component of a VNF, and VNFC Instances (VNFCIs) are the executing constituents which make up a VNF Instance. In order to instantiate the virtual network function defined by a VNF, VNF Manager create one or more VNFCIs, where each VNFCI is in its own virtualisation container. These VNFCIs, between them, provide the functionality of the VNF, and expose whatever interfaces are provided by that VNF. The requirements for initial deployment state are described in the VNF Descriptor (VNFD) [i.3], including the connections between VNFCIs which are internal to the VNF, and not visible to external entities at the VNF level. Post-deployment operation capabilities, such as migration of the VMs containing VNFCIs, scale up, scale down, scale in, scale out, changes to network connections, etc., are also described in the VNFD. VNFCIs may have some ability to make changes to the connections between themselves, but those which require changes to the NFVI (other than when exposed, for instance, via hardware drivers) are accomplished by the NFV Management and Orchestration. These entities may derive them automatically, or via directions or requests from the VNF, via the VNFM for that VNF.

Each VNF has exactly one associated VNF Descriptor (VNFD) (see clause 8.2).

The functional and non-functional assurances (e.g. performance, security, reliability) that a VNF Provider makes in the VNFD can only be met as long as the VNF's integrity is maintained. In other words, these assurances would be void if one or more of the VNF's software components were to be replaced by any other components. Therefore, a VNF can generally be assumed to be packaged by a single VNF Provider. If the VNF Provider integrates third-party software components into their VNFs' implementation it is their responsibility to provide a VNF package.

This does not in any way preclude the reuse of software components that are common to two or more VNFs. To reuse such a common component, it would have to be factored out of the original VNFs into a new type of VNF with well-defined interfaces. This use case will be described in a later clause.

One of the goals of NFV is to be able to create a structure where key network capabilities can be decoupled as VNFs from the infrastructure on which the VNFs execute so the VNFs can scale when necessary. In order to properly virtualise these types of capabilities they need to be well understood, defined, and catalogued for proper inclusion in service orchestrations.

In order to meet the above design objectives the following factors need to be taken into account:

For that two factors need to be taken into account:

- functionality - commonly acceptable functionality is either defined or described in the industry (e.g. 3GPP MME); and
- its atomicity has to be guaranteed to ensure that it can be developed, tested and deployed as a separate virtualised network function.

4.3 Interfaces

4.3.1 VNF Interfaces and NFV Architectural Framework Reference Points

An interface is a point of interaction between two entities. The entities can be software and/or hardware services and/or resources. Software interfaces separate the specification of how software entities communicate from the implementation of the software. They are used to create well-defined points of interaction between software entities, and restrict communication between software entities to those interaction points. Hardware interfaces are beyond the scope of the present document.

NOTE 1: ETSI GS NFV 002 [i.4] uses "reference points", not "interfaces". Briefly, a reference point is an architectural concept that defines and exposes an external view of a function block. For the present document, interfaces are more natural terminology, since the point of the interface is to allow communication between two entities.

The present document identifies 5 types of interfaces relevant to the VNF software architecture. Figure 3 shows a set of interfaces as identified in the architectural framework defined in ETSI GS NFV 002 [i.4], and how these five types of interfaces indicated in figure 2 (i.e. SWA-1 - SWA-5) are related to it.

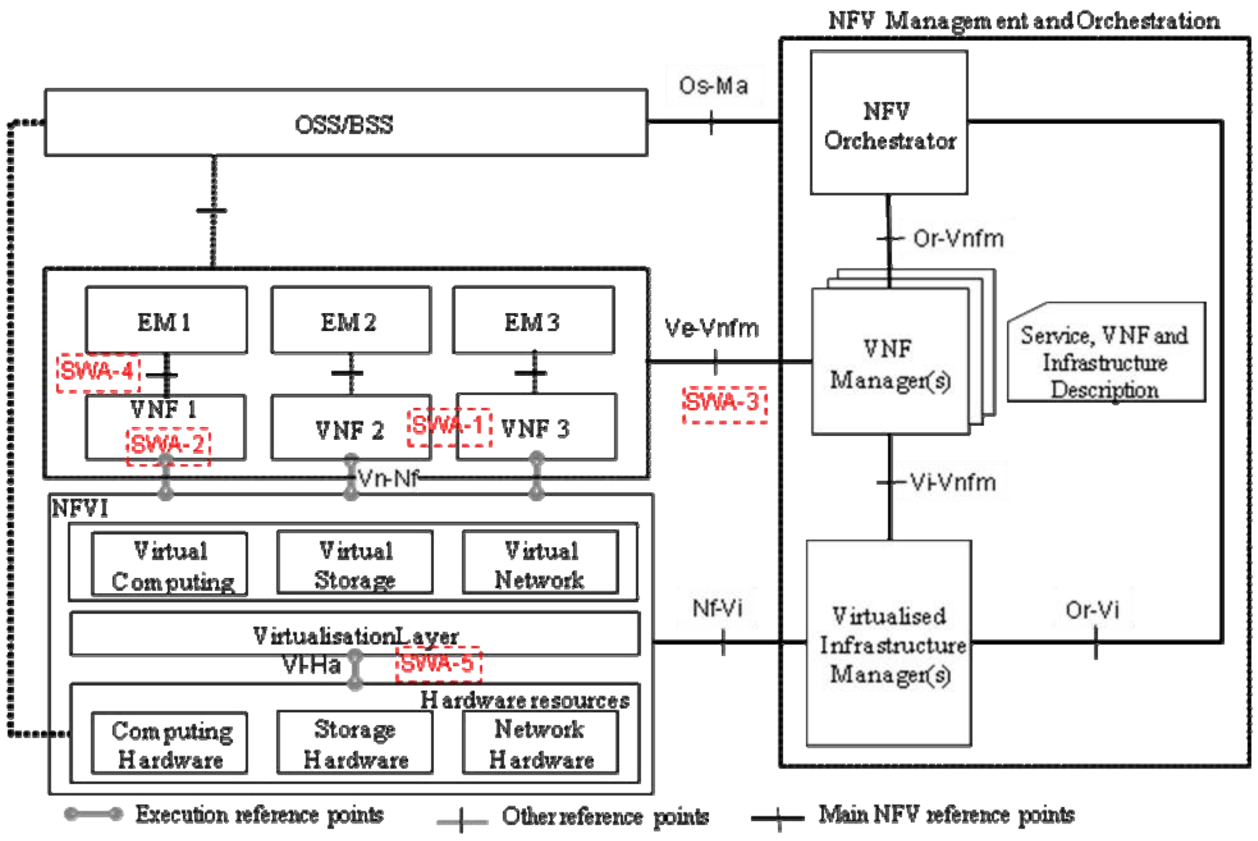

Figure 3: NFV Architectural Framework (5 types of VNF interfaces)

Figure 4 shows an evolving view of the NFV architecture framework, providing more details on the aspects that are relevant to the VNF software architecture.

It also clarifies that when two or more VNFCs are instantiated, each VNFC instance is hosted inside its own unique virtualisation container, and the NFVI uses multiple, separate instances of the SWA-5 (Vn-Nf) interface to provide access to the hosted execution environment of each separate VNFC instance.

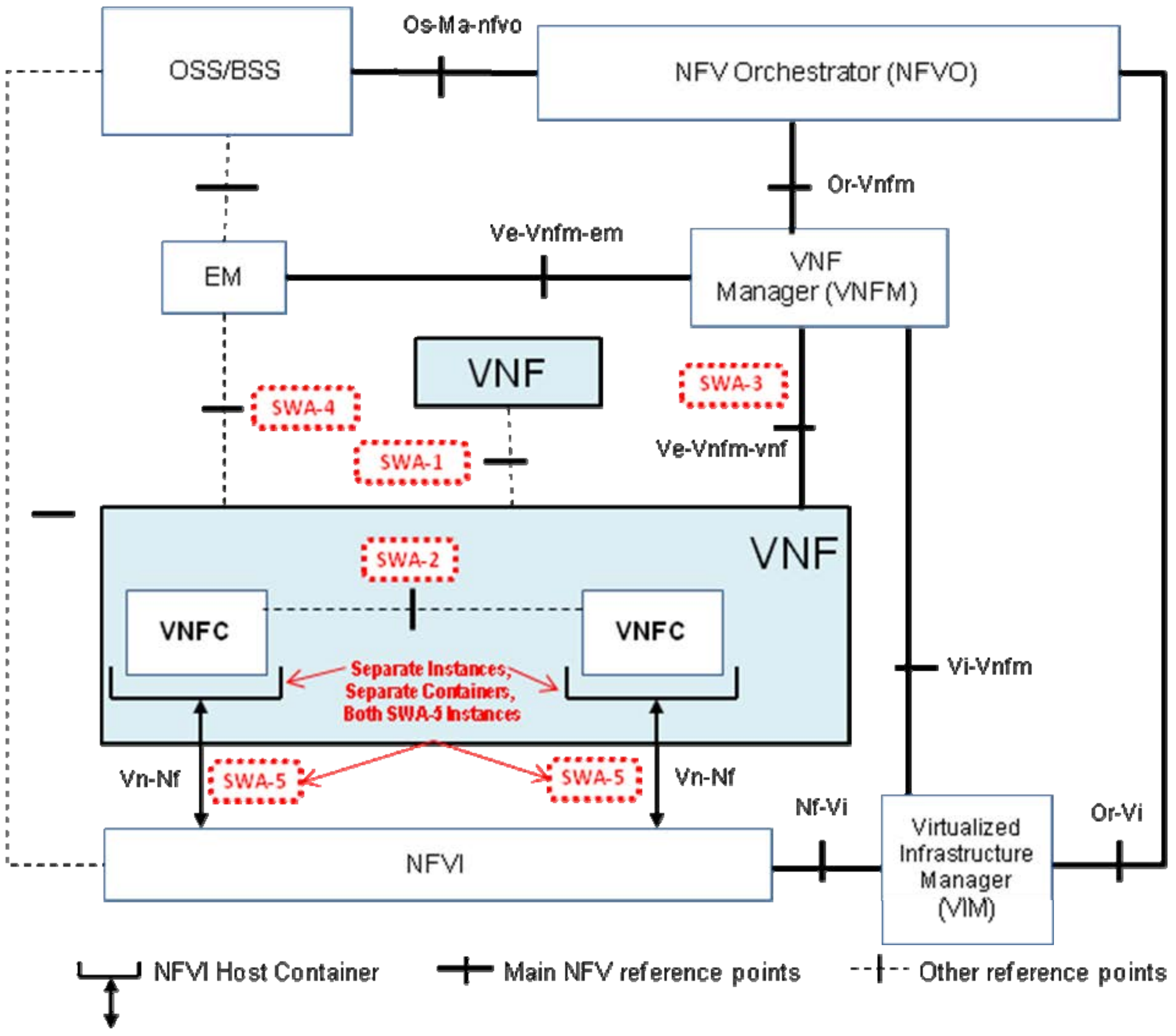

Figure 4: VNF interfaces and NFV Architectural Framework reference points

The EM, the VNF Manager, as any other entity identified by the NFV architecture framework, may be virtualised or not.

NOTE 2: SWA-3 corresponds to Ve-Vnfm-vnf [i.3].

4.3.2 SWA-1 Interfaces

SWA-1: This is a well-defined interface enabling communication between various network functions within the same or different network service. They may represent data and/or control plane interfaces of the network functions (VNF, PNF). The SWA-1 interface is between two VNFs, a VNF and a PNF, or between a VNF and an End Point [i.4] as shown in figure 5. A VNF may support more than one SWA-1 interface.

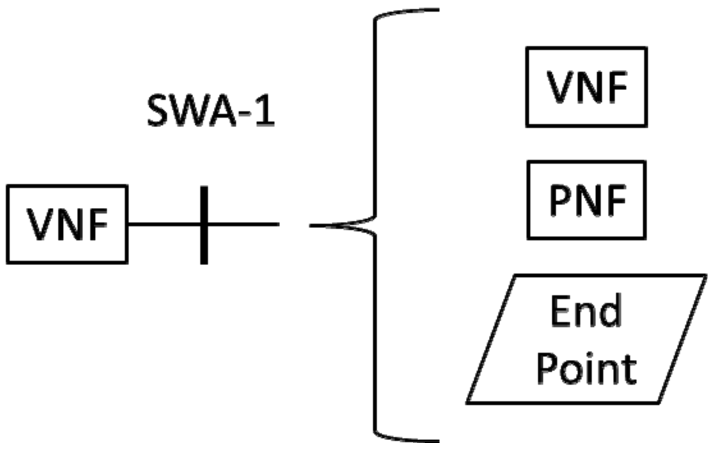

Figure 5: SWA-1 Interfaces

The Network Operator or Service Provider composes the network services. In order to compose a viable network service, the Network Operator or Service Provider requires that the VNFD of a given VNF provide sufficient information about the SWA-1 supported by that VNF to determine compatibility with a different VNF obtained from a different VNF Provider.

SWA-1 interfaces are logical interfaces that primarily make use of the network connectivity services available at the SWA-5 interface.

4.3.3 SWA-2 Interfaces

The SWA-2 interfaces refer to VNF internal interfaces, i.e. for VNFC to VNFC communication. These interfaces are defined by VNF Providers, and are therefore vendor-specific. Those interfaces typically place performance (e.g. capacity, latency, etc.) requirements on the underlying virtualised infrastructure, but are not visible from the perspective of a VNF user. The details of this interface are out of scope of the present document with the exception of providing requirements to NFV management orchestration functions (e.g. VNF Manager, NFVO and VIM) and the NFV Infrastructure.

The SWA-2 interfaces leverage underlying communication mechanisms implemented over the NFV Infrastructure.

SWA-2 interfaces are logical interfaces that primarily make use of the network connectivity services available at the SWA-5 interface.

Optionally, when two VNFC instances are deployed on the same host, other technologies may be used to support better latency constraints including:

- Serial Bus like technologies: A channel-based communication mechanism (CPU level speed-controlled vs. NIC schedule-controlled) that allows inter-partition (i.e. the VNFCs) communication.
- Shared Memory Access: A mechanism that enables one VNFC to access shared memory with a neighbouring VNFC.

NOTE: The use of Shared Memory Access has inherent security risks, and should only be used under controlled circumstances. In addition, since this function inherently restricts flexibility in deployment, it is not a mandatory interface or feature as indicated above.

4.3.4 SWA-3 Interfaces

The SWA-3: interfaces the VNF with the NFV management and orchestration, specifically with the VNF Manager.

- Role - The Management Interfaces are used to perform the life cycle management (e.g. instantiation, scaling, etc.) of one or more VNFs.
- Interconnection attributes - SWA-3 interfaces may use IP/L2 connectivity.

4.3.5 SWA-4 Interfaces

The interface SWA-4 is used by the EM to communicate with a VNF. This management interface is used for the runtime management of the VNF according to the Fulfilment, Assurance, and Billing FAB (e.g. as defined in the eTOM [i.22]) and FCAPS (e.g. as defined in ITU-T TMN recommendations M.3010 [i.23], M.3400 [i.24] and X.700 [i.25]) network management models and frameworks.

4.3.6 SWA-5 Interfaces

SWA-5 corresponds to VNF-NFVI interfaces: This is a set of interfaces that exist between each VNF and the underlying NFVI. Different types of VNFs exist, as well as collections of VNFs, such as VNF Forwarding Graphs and VNF Sets. Each of these different VNF relies on a potentially different set of NFVI resources (e.g. resources for networking, compute, storage, and hardware accelerators) that are provided at the SWA-5 interface. The SWA-5 interface provides access to a virtualised slice of the NFVI resources allocated to the VNF, i.e. to all the virtual compute, storage and network resources allocated to the VNF. Thus the SWA-5 interface describes the execution environment for a deployable instance of a VNF.

The SWA-5 interfaces are an abstraction of all the sub-interfaces between the NFVI and the VNF itself. Each sub-interface has a specific set of use(s) and role(s), and type of inter-connection attribute(s). This is shown in figure 6. The SWA-5 sub-interfaces are:

Generic compute functionality:

1) Role - The NFVI provides an interface to access generic compute functionality.

2) Interconnection attributes - These sub-interfaces have CPU-dependent attributes.

Specialized function:

1) Role - A Specialized function provides the VNF with non-generic compute functionality or expanded generic compute functionality. These functions can vary from a Video or Voice Gateway card, to specialized memory and or routing application.

2) Interconnection attributes - The SWA-5 specialized interface(s) are strictly dependent upon the technology being implemented and the interface types associated with that standard. Generically the interfaces should have their own identity, and mapping capabilities.

Storage:

1) Role - The NFVI provides a storage interface that can support storage operations on any granularity including block, file, or object storage.

2) Interconnection attributes - Physical storage in the NFVI may be provided locally or remotely.

3) Storage operations - In order to access the storage capabilities of the NFVI, the VNF developer may use common storage operations as shown in table 1. The execution environment provided by the NFVI may implement these operations using a number of different device drivers to mask technology differences.

Table 1: Storage Operations

Item	Operation
Storage Device	Mount/Unmount
Directory	Create/Delete
File	Open/Close/ Read/ Write

Network I/O:

1) Role - The SWA-5 Network I/O interface(s) provide the VNFC/VNF instanceswith network connectivity services

   a. Layer 2 services (e.g. E-LAN, E-Line, E-Tree) based on the Ethernet switching network infrastructure or on BGP Ethernet VPNs (EVPN).

b. Layer 3 services (directly based on the L3 infrastructure or on Layer 3 VPNs)

NOTE: If the VNFC/VNF instances communicate using IP, either a Layer 2 or Layer 3 service may be used depending on the granularity required.

2) Multiple interfaces for both redundancy and segmentation are normal considerations, along with hybrid Network Interface Cards (NICs) that provide multiple ports. Each virtual NIC has a driver for the supported OS/VNF.

3) Interconnection attributes - Each (can be multiple) SWA-5 Network I/O interface has NIC-level attributes, and needs to be separately map-able to a network segment, and also map-able in pairs for redundancy with network functions such as Link Aggregation Groups (LAG).

The Network I/O sub-interface maps to the [Vn-Nf]/VN reference point specified in ETSI GS NFV INF 001 [i.1]. All other sub-interfaces map to the [Vn-Nf]/VM reference point specified in ETSI GS NFV INF 001 [i.1].

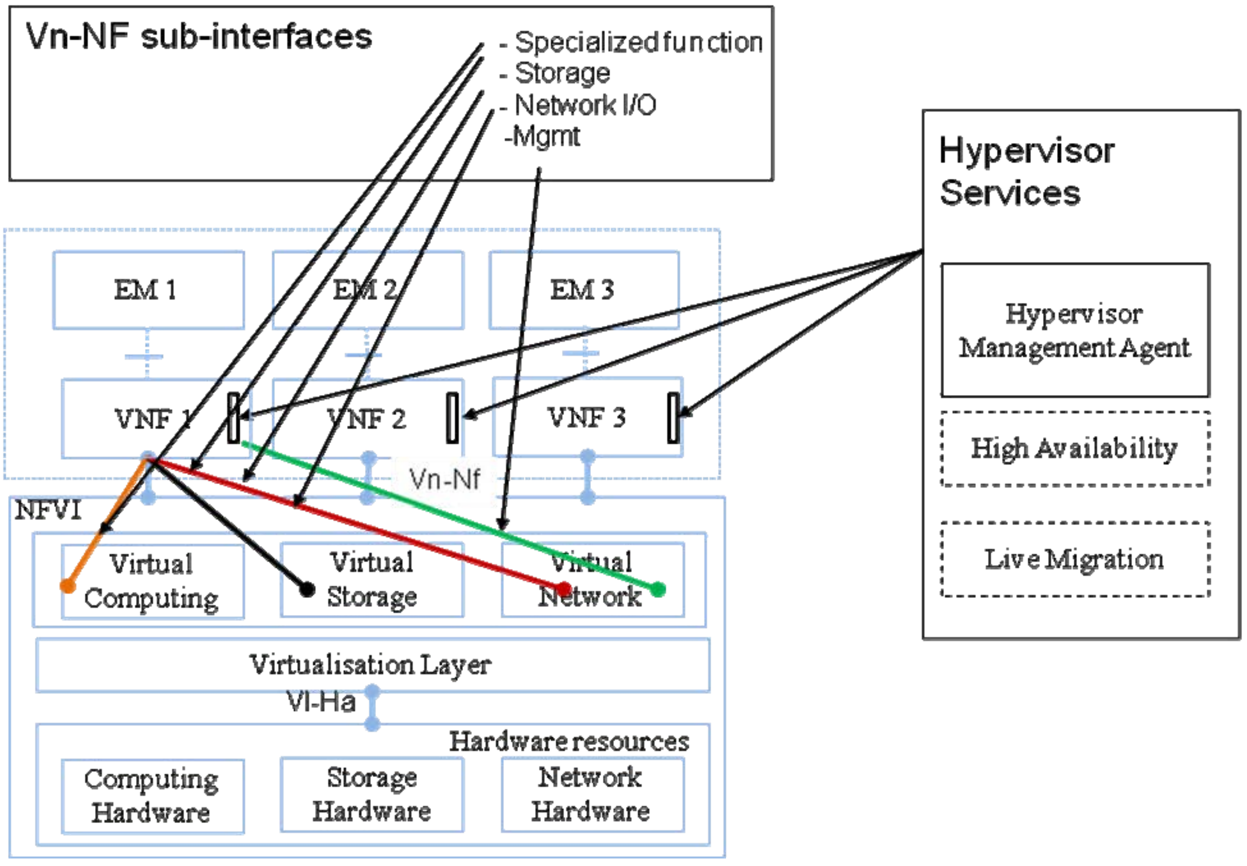

Figure 6: Detail view of the Sub-Interfaces of the SWA-5 (Vn-NF) Interfaces

5 VNF Design Patterns and Properties

5.1 VNF Design Patterns

This clause is describing common patterns in VNF design and operations. The goal is to capture all practically relevant points in the design space, from which requirements on the VNFD, the NFVO and NFVI can be derived.

5.1.1 VNF Internal Structure

A VNF may be composed of one or multiple components, called VNFC. A VNFC in this case is a software entity deployed in a virtualisation container, see figure 7.

A VNF realized by a set of one or more VNFCs appear to the outside as a single, integrated system.

The same VNF may be realized differently by each VNF Provider. For example, one VNF Provider may implement a VNF as a monolithic, vertically integrated VNFC, another VNF Provider may implement the same VNF using separate VNFCs, say one for the control plane, one for the data plane and one for element management.

VNFCs of a VNF are connected in a graph. For a VNF with only a single VNFC, that internal connectivity graph is the null graph.

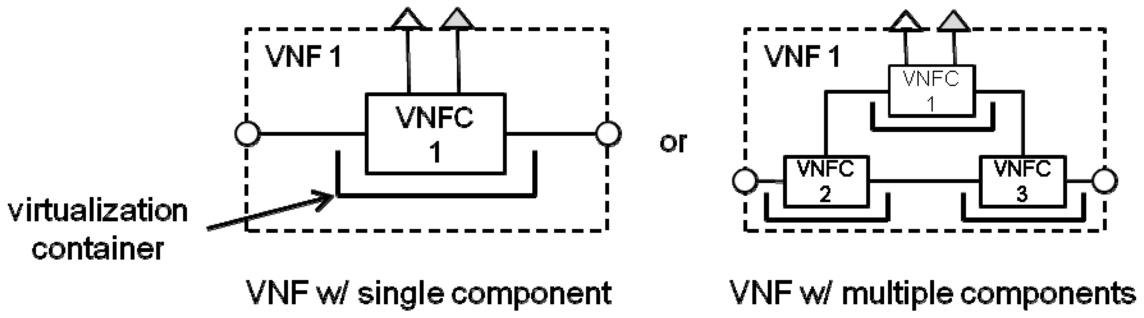

Figure 7: VNF Composition

5.1.2 VNF Instantiation

Each VNFC of a VNF is either parallelizable or non-parallelizable:

- If it is parallelizable, it may be instantiated multiple times per VNF Instance, but there may be a constraint on the minimum and maximum number of parallel instances.
- If it is non-parallelizable, it shall be instantiated exactly once per VNF Instance.

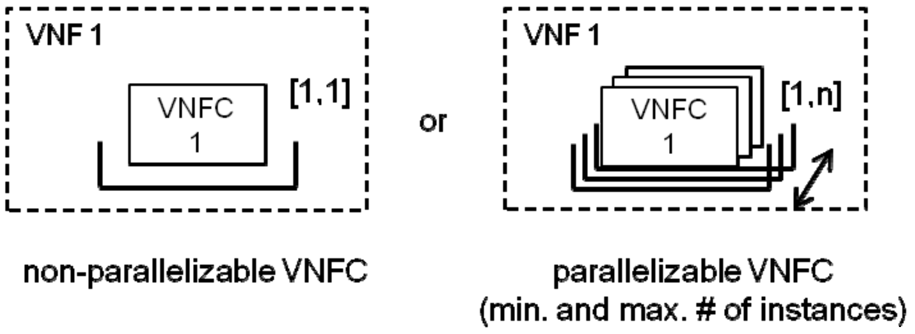

Figure 8: VNF Instantiation

5.1.3 VNFC States

Each VNFC of a VNF may need to handle state information. Each VNFC of a VNF is either stateless or stateful. A VNFC that does not have to handle state information is a stateless VNFC. A VNFC that needs to handle state information may be implemented either as a stateful VNFC or as a stateless VNFC with external state (state data is held in a data repository external to the VNFC).

- Statefulness will create another level of complexity, e.g. a session (transaction) consistency has to be preserved and has to be taken into account in procedures such as load balancing.
- The data repository holding the externalized state may itself be a stateful VNFC in the same VNF.
- The data repository holding the externalized state may be an external VNF.

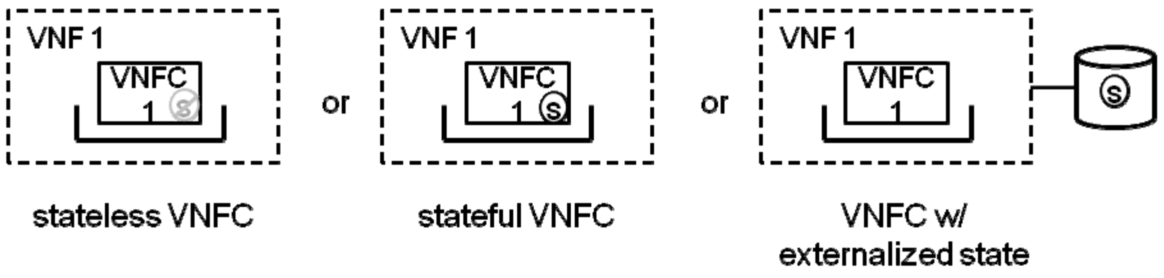

Figure 9: VNF States

5.1.4 VNF Load Balancing Models

There are different types of load balancing, typically 4 models are identified:

1) VNF-internal Load Balancer:

- 1 VNF instance seen as 1 logical NF by a Peer NF. The VNF has at least one VNFC that can be replicated and an internal load balancer (which is also a VNFC) that scatters/collects packets/flows/sessions to/from the different VNFC instances. If the VNFCs are stateful, then the LB shall direct flows to the VNFC instance that has the appropriate configured/learned state.
- Examples: VNF Provider specific implementation of a scalable NF.
- Single VNF Provider solution (per definition of VNFCs).
- The VNFM instantiates the LB, which may itself be a VNFC.

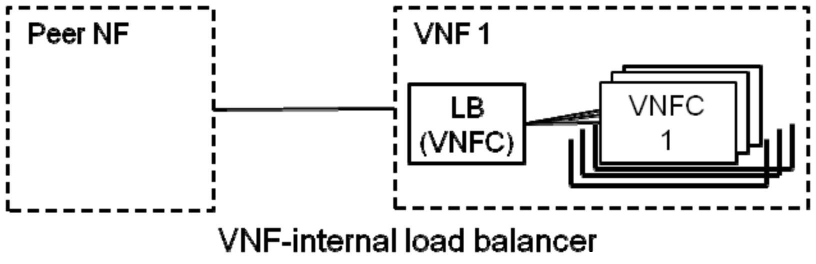

Figure 10: VNF Internal Load Balancer

2) VNF-external Load Balancer (analogous to 1:1 (drop-in) replacement of existing NFs):

- N VNF Instances seen as 1 logical NF by a Peer NF. A load balancer external to the VNF (which may be a VNF itself) scatters/collects packets/flows/sessions to/from the different VNF instances (not the VNFCs!).
- Examples: Application Delivery Controller (ADC) type LB or Direct Server Return (DSR) type LB in front of web-server.
- VNFs may be of different VNF Providers, e.g. to increase resilience.
- If the VNF supports this model, the NFVO may instantiate it multiple times and add a LB (V)NF in front of this pool of VNF instances.
- If the VNFs contain state, then the LB VNF shall direct flows to the VNF instance that has the appropriate configured/learned state.

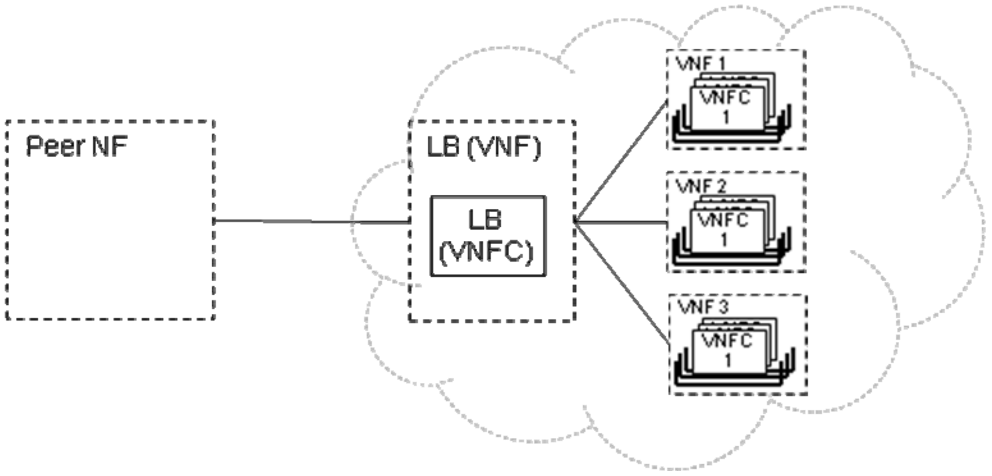

Figure 11: VNF External Load Balancer

3) End-to-End Load Balancing:

- N VNF instances seen as N logical NFs by a Peer NF. The Peer NF itself contains load balancing functionality to balance between the different logical interfaces, see figure 12. For example:
    - In 3GPP, S1-flex interface between eNBs (= Peer NF) and MME, S-GW.
    - Client-side (DNS-based) load balancing between web servers.
- VNFs may be of different VNF Providers, e.g. to increase resilience.
- If the VNFs contain state, then the LB NF shall direct flows to the VNF instance that has the appropriate configured/learned state.
- The NFVO may instantiate multiple VNFs, but does not instantiate a LB.

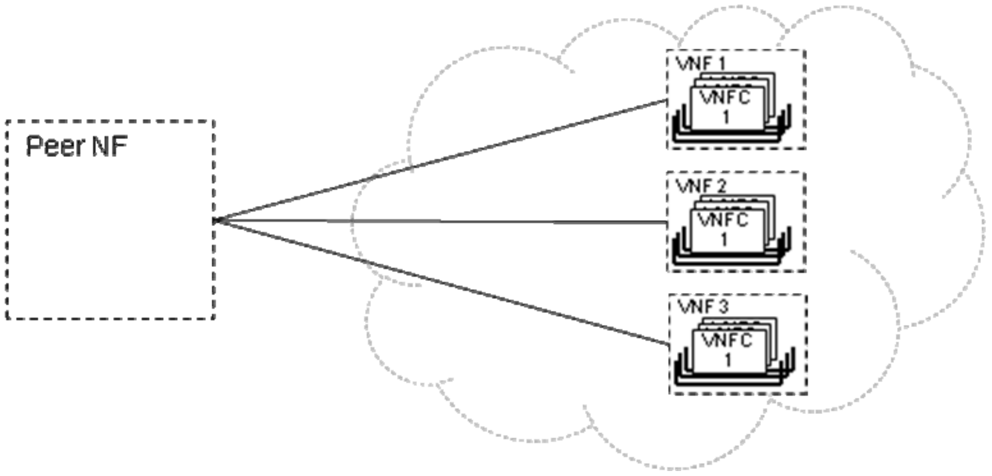

Figure 12: VNF E2E Load Balancing

4) Infrastructure Network Load Balancer

- VNF instances seen as one logical NF by a Peer NF. A load balancer (may be in a hypervisor vSwitch, an OS vSwitch, or a physical box) provided by the NFV Infrastructure that scatters/collects packets/flows/sessions to/from the different VNF instances. Examples: NFV Infrastructure providing load balancing for several Firewall instances.
- VNFs may be of different VNF Providers, e.g. to increase resilience.

- If the VNF and the NFVI supports this model, the NFVO may instantiate it multiple times and configure a load balancer in the NFV Infrastructure that connects to this pool of VNF instances to perform load balancing.

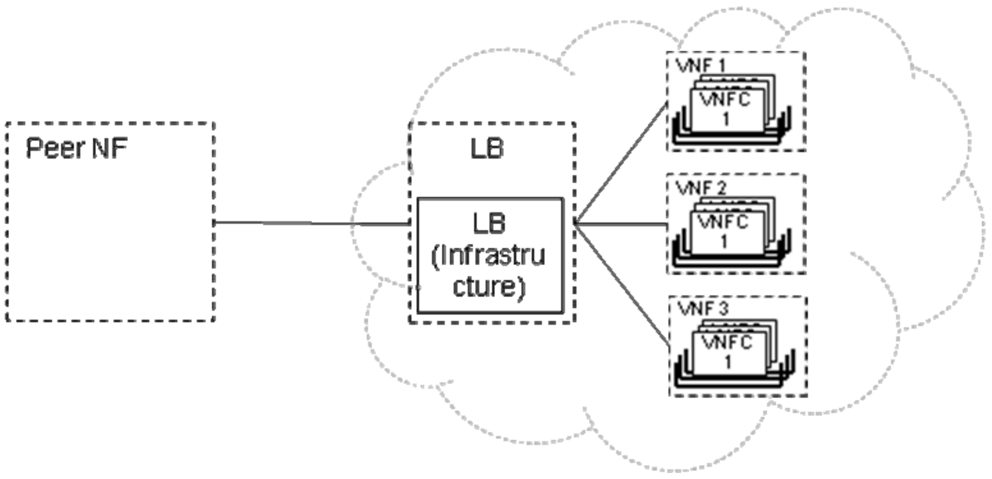

Figure 13: Infrastructure Network Load Balancer

5.1.5 VNF Scaling Models

There are different types of VNF scaling, typically 3 basic models have been identified:

1) Auto scaling:

   - The VNF Manager triggers the scaling of VNF according to the rules in the VNFD, e.g. based on monitoring of resource utilization of the VNF's VMs, upon events received from the VNF, the EM, the VIM, or locally generated. Both scale out/in and scale up/down may be supported.

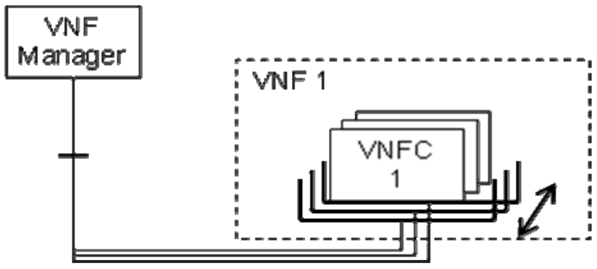

Figure 14: Auto Scaling trigger by the VNF Manager

2) On- demand scaling (from VNF or EM):

   - On-demand scaling, in which a VNF Instance or it EM monitor the states of the VNF Instance's constituent VNFC Instances and trigger a scaling operation through explicit request to the VNF Manager to add or remove VNFC instances (scale out or scale in, respectively) or increase or decrease resources of one or more VNFC instances (scale up or scale down, respectively).

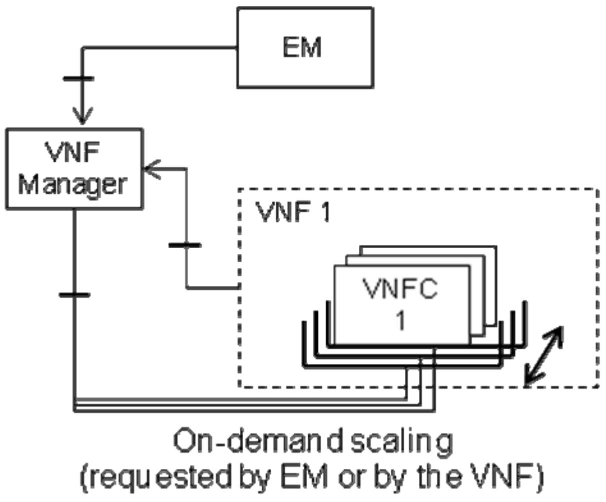

Figure 15: On-Demand Scaling (ex: scale out/in)

3) Scaling based on a management request.

Manually triggered scaling (e.g. by NOC operators) or OSS/BSS triggered scaling according to the rules in the VNFD by issuing requests to the NFVO via an appropriate interface. Both scale out/in and scale up/down may be supported.

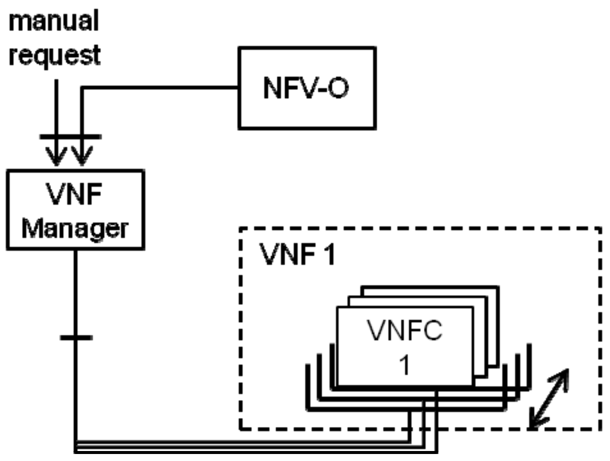

Figure 16: Scaling based on management request

5.1.6 VNF Component Re-Use

Different models have been studied regarding component re-use but only one case has been agreed as relevant for ETSI NFV.

The first model that has been considered can be described as follows:

- Assume two VNFs X and Y containing VNFCs B1 and B2, respectively. B1 and B2 provide the same functionality.
- X and Y may be the same or different VNFs.
- X and Y may be from the same or a different VNF Provider.
- B1 and B2 may actually be identical, e.g. bought-in from a 3rd party.

Whether a VNF's software components of a VNF are being reused in one of these cases or not is a VNF Provider design choice and is out of the scope of the present document. It should therefore not be referred to as "Component Reuse" in the present document, see figure 17.

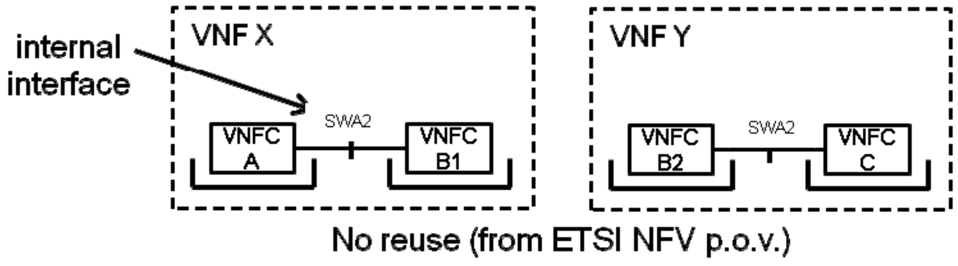

Figure 17: VNFC Re-Use

The second model consists in having an instance of VNF X and an instance of VNF Y "share" an instance of a VNFC B*. It is not a valid model, because it is not clear to which VNF the VNFC B* is allocated and because it violates the capsulation of the VNF by making an internal interface external, see figure 18.

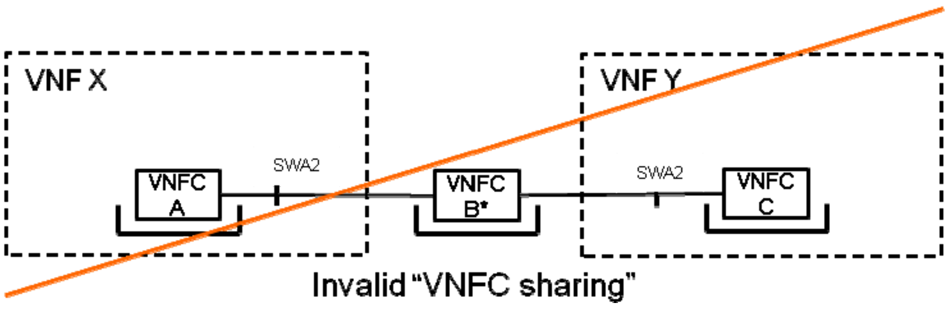

Figure 18: Invalid VNFC Sharing

The third model refer to the scenario where the common VNFC B* is factored out of VNFs X and Y and turned into a proper VNF (by adding a VNFD to it) that may or may not come from a different VNF Provider. All of those VNFs are then handled like any other VNF:

- VNFs X and Y do not remain the same function as a result, but become new functions (here: A and C).
- The internal interface (2) becomes an external interface (1).
- Most importantly, though, the SLAs of VNFs X and Y towards the NFVO change: A and C are no longer responsible for the performance (or lack of performance) of VNF B.

This is the only valid model for "Component Reuse" in the context of ETSI NFV.

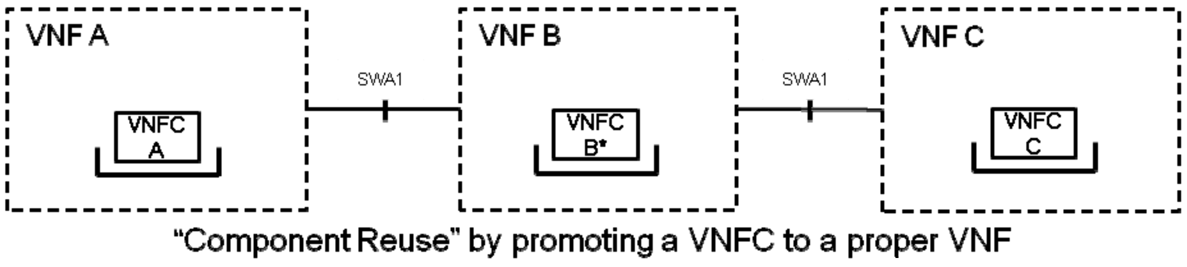

Figure 19: Proper "Component Re-use"

5.2 VNF Update and Upgrade

5.2.1 VNF Update and Upgrade Overview

As any product, a VNF requires continuous development of new functionality and maintenance of the existing functionality. Changes introduced during development and maintenance are often categorized into updates and upgrades. Although there is no formal definition of an update and an upgrade, a very general way of distinguishing them can be:

- A VNF update does not introduce new functionality and/or new interfaces.
- A VNF upgrade might introduce new functionality and/or new interfaces.

The key difference is that VNF upgrades may require planning on network service level, while VNF updates can be deployed without coordination with other VNFs participating in the same VNFFG.

The methods of VNF updating and VNF upgrading differ from vendor to vendor. For a specific vendor the method may be the same or different for VNF updates and VNF upgrades.

5.2.2 VNF Update & Upgrade Requirements for VNF Provider

The following requirements are VNF Update and Upgrade for VNF Provider:

1) The VNF Provider shall provide an automatic procedure that executes the VNF Instance update/upgrade. That procedure should be a part of the VNF Package.
2) The procedure shall control the progress of the VNF Instance update/upgrade including requesting virtual resources from the NFV management and orchestration.
3) The procedure shall support the roll-back of the VNF Instance update/upgrade including returning virtual resources obtained from the NFV management and orchestration.

5.3 VNF's Properties

The following clauses list NF properties of many dimensions. Each clause focuses on one property and creates a list of categories for classification of the NFs according to this property.

Many of the listed properties will need to be reflected in a descriptor made available to management and orchestration functions, but some properties are just necessary here to understand the software architecture.

5.3.1 Hardware Independence

Some NFs might not fully realize hardware independence for various reasons (implementation history, performance/latency aspects, special requirements). Therefore this clause categorizes the NFs in the way they can realize hardware independence:

- COTS-Ready:
  - The NF fully realizes independence from the physical infrastructure, allowing the network function to be run on any appropriate physical infrastructure.
  - Examples: Typically session controllers are implemented COTS-Ready.
- Partly COTS-Ready:
  - The NF in some of its VNFCs fully realizes independence from the physical infrastructure, while some VNFCs can run only on a specified type of hardware.
  - Examples: MRF, see use case in the appendix. Here a VNFC requires special DSP card, but other VNFC's are running on COTS, so the VNF is partly COTS ready.

- Hardware dependent:
  - The NF requires a specified type of hardware for all its components (it is still a PNF, so not fully in scope of NFV).
  - Examples: A DPI implementation could require a specific data acquisition card and so be hardware dependent. Also UDP fire walls, and gateway devices in many cases are not COTS-Ready.

Every NF will belong to one of the three categories. NFs of the first two categories may also have VNFCs that require direct hardware access via standardized interfaces, but this makes them independent of different COTS hardware implementations.

5.3.2 Virtualisation and Container Awareness

Some NFs might not be able to run in any virtualised environment for various reasons (implementation history, performance/latency aspects, special requirements). Some NFs that are able to run in a virtualised environment in the form of a VNF are designed for OS containers, while other are designed for hypervisor virtualisation. These categories describe the NF's software relation to the various container and virtualisation technologies. In most cases a NF will belong to exactly one category, but some categories can also be combined.

- Hipervisor agnostic:
  - The network function is able to run in a virtual machine/hypervisor environment. Its software has no dependency to the virtual machine/hypervisor environment, so it can run unchanged on different hypervisors - or also on native hardware (typically COTS hardware).
- Hypervisor dependent:
  - The network function is able to run in a virtual machine or some other abstraction mechanism between the logical environment seen by the network function and the actual physical infrastructure.
  - But the VNF implementation is dependent on the virtual machine/hypervisor environment it is running in.
- Operating System Containers:
  - The NF software is designed for use with OS container technology for its deployment. Note that there is always some dependency between application and OS in this technology.
- Higher layer container technologies:
  - The NF software is able to run on some higher layer container technology such as Java virtual machines.
- Not virtualised and no container technology:
  - The network function cannot run in a virtual machine/hypervisor environment and does not use other container technologies. Therefore it could be seen outside the scope of NFV. It is a PNF.
- Partly virtualised:
  - Some components of the NF can be virtualised, some may be virtualisation aware, some not virtualised.

5.3.3 Elasticity

This property describes the ability of a VNF to define and perform elastic operations as well as the type of scaling it can use (up/down, out/in). Every VNF will belong to exactly one of these categories.

- No elasticity:
  - The VNF requires a fixed set of resources that cannot be changed.

- Elasticity by scaling up/down only:
  - The NFV framework can increase or decrease the size, performance or bandwidth of the virtual resources.
- Elasticity by scaling out/in only:
  - The VNF is able to perform scaling operations by separately adding/removing instances of specified VNFCs.
- Elasticity in either dimension:
  - The VNF has VNFCs that may be scaled out/in, up/down or both.

5.3.4 Void

5.3.5 VNF Policy Management

This property describes the ability of a VNF to support dynamic rule-based provisioning and management needed for automated operational tasks (e.g. scale-in, scale-out, or migration based on threshold crossing). There are some or more policies for each automatic operation task of VNF. Every VNF will belong to exactly one of the categories below.

- Fully policy based VNF:
  - NFV orchestration and management provides full set of provisioning and management policies for this VNF.
- Not policy based VNF:
  - NFV orchestration and management will not provide any provisioning and management VNF policies for this VNF. VNF will provide policy management by itself.

5.3.6 Migration operations

This property describes the ability of a VNF to perform migration operations of its resources. A VNF can belong to the first category or allow any of the other attributes in combination.

- No live migration supported:
  - The VNF does not support that its resources are migrated during their time of assignment.
- Live migration supported:
  - The VNF allows its resources to be migrated to a different NFVI resources using hypervisor technology. Typically this includes specifying live migration policies, e.g. performance parameters.
- Migration partially supported:
  - Some VNFC support migration while others do not.
- Other migration mechanisms:
  - The VNF allows other mechanisms (e.g. shutdown and restart on different hardware) to change the resource assignments dynamically. These mechanisms might make use of the VNF's internal redundancy schemes.

EXAMPLE: If a VNF internally uses 1+1 redundancy, so one VNFC always is active and one standby, it is possible to migrate the VNFCs by shutdown of the standby and restart on a different resource and then performing a switchover before migrating the second VNFC.

5.3.7 VNF State

This property describes whether the VNF needs to keep state information. Every VNF will belong to exactly one of these categories.

- Stateful operation:
    - The definition of the VNF's task or interfaces require that the VNF needs to keep state information.
- Stateless operation:
    - The VNF does not require to keep state information.

NOTE: In the last case, It may use an external state repository.

5.3.8 VNF Internal Structure

This property describes the deployment structure of the VNF. Every VNF will belong to exactly one of these categories.

- Simple: the VNF contain a single VNFC. In this case, SWA-2 type interfaces are not required (refer to clause 4.4.3).

NOTE: The VNFC could contain clearly separated functions, which the VNF Provider is able to reuse internally.

- Structured: the VNF consists of multiple VNFC's that typically have some dependency on each other, so they may need specific rules about their location/topology. Inter-VNFC communication is supported by means of SWA-2 type interfaces (refer to clause 4.4.3).

5.3.9 Reliability

The Reliability architectural patterns, their dependencies and requirements towards VNFs are documented in [i.9].

VNF Fault Management is somehow related to reliability in the sense that reliability mechanisms are dependent on fault detection and fault reporting mechanisms. These dependencies are documented in [i.9].

5.3.10 Location Awareness

This property describes dependencies of the VNF or some of its components on a position in the topology or geography (see topology requirements in clause 8).

5.3.11 Application Management

This property describes what type of application management needs to be used for the VNF. A VNF can belong to the first category indicated below or allow a combination of the other categories.

- No management:
    - The VNF does not require management operations during its lifetime. All configuration shall be done during installation.
- Standardized management interface:
    - The VNF provides a standard management interface to be used by a separate Element Management (EM), or other OSS/BSS software packages.
- Proprietary management interface:
    - The VNF provides a non-standard OAM interface to be used by a separate Element Management by the same VNF Provider.
- Multiple service providers' management interfaces.

5.3.12 Diversity and Evolution of VNF Properties

In a VNF with multiple VNFCs, each VNFC may have different properties. For example, one VNFC may be stateful and non-parallelizable, while another VNFC may be stateless and parallelizable.

The same VNF may be implemented with different properties, and it is conceivable that a single vendor's VNF implementation evolves over time to exhibit different properties. For example, a VNF may evolve from single VNFC with stateful, non-parallelizable properties to a multi-VNFC in which some or all VNFCs have externalized state and are parallelizable.

5.4 Attributes describing VNF's Requirements

5.4.1 VNF Topological Characteristics

The deployment configuration and operational behaviour of a VNF shall be described according to a template called Virtualised Network Function Description (VNFD) [i.3]. Deployment behaviour defines the state and environment for a VNF to be deployed, where as operational behaviour defines the needed functions for a VNF to be operated and managed properly.

The template captures the general characteristics of each VNF and is used to on-board the VNF, in order to support on-demand instantiations of the VNF in an operator's network.

The VNFD is composed of the following main information elements groupings:

- VNF identification data, including:
  - Data to uniquely identify the VNF vendor/provider.
  - Type and description of the VNF, which help to identify the Network Function that is implemented as a Virtual NF, and enable interoperability of VNFs manufactured by different VNF Providers.
  - Version.
- VNF specific data, including:
  - Specific VNF configuration data.
  - Connectivity requirements and inter-dependencies of VNFCs (see clause 5.4.1.1.3).
  - VNF lifecycle workflow scripts (see clause 5.4.1.2.1).
  - Deployment flavours, specifying how many instances of each VNFC type and version to be instantiated based on service KPIs.
  - Deployment constraints (see clauses 5.4.1.1.4 and 5.4.1.1.5)
- VNFC data (see clause 5.4.1.1.1), including:
  - Type and identification, uniquely identifying the VNFC type.
  - Specific VNFC configuration data and scripts.
  - Deployment constraints (see clauses 5.4.1.1.4 and 5.4.1.1.5)
  - Virtualisation container files/images references, including the possibility to define packages of:
    - VNFC binaries plus operating system;
    - Empty operating system; and/or
    - Empty virtualisation container (i.e. unloaded operating system).
- Virtualised resource requirements (see clause 5.4.1.1.2), including:
  - Compute resources, e.g. virtual CPU and virtual RAM assigned to the virtualisation container.

- Storage resources, e.g. size of the virtual disk assigned to the virtualisation container.
- Network resources, e.g. number and type of virtual NICs that are assigned to the virtualisation container, including also requirements for network bandwidth.

5.4.1.1 Deployment Behaviour

5.4.1.1.1 Virtualisation containers

A VNF may be composed of one or more components, called VNFC, with each VNFC instance deployed on one virtualisation container (e.g. VM). Hence the description shall include the number of VNFCs required for this VNFs instantiation, including the corresponding Virtualisation container files/images references.

For each sub-component, VMs with different characteristics may also be specified to indicate the extent of scaling up/down a particular sub-component of the VNF. This information may be used at run-time for automated scaling up/down.

5.4.1.1.2 NFVI Resources

The VNF design may have particular characteristics to be met by NFV Infrastructure resources. The description of VNF in terms of required computational, networking and storage resources per VNF sub-component (if applicable) is used to determine the type of virtualisation container(s) (e.g. VM(s)) to be allocated.

The NVFI resource requirements for a VNF may be expressed as targets in terms of CPU processing power, storage size, connectivity latency, etc. or may be captured as a NFV Infrastructure "class of service" that meets those characteristics and maps to a particular category of pre-configured VMs and connectivity in the operator's domain.

5.4.1.1.3 Components and Relationship

This requirement defines different components of a VNF and their relationship between each other. The integrity of the entire VNF configuration is ensured by connecting all its sub-components as per the relationship between them. The relationship may be defined in terms of redundancy models (simplex, active-standby, active-active, n+k, etc.), functional dependencies (hosted-on, connect -to, etc.), etc.

5.4.1.1.4 Location

The description of the VNF may include deployment constraints on the physical location of a VNF or of one of its VNFC. For example, there might be a constraint for redundancy that dictates how many instances of a VNF can be collocated in the same location.

5.4.1.1.5 Other constraints

The description of the VNF may impose other constraints, e.g. regarding degree of isolation (VNF may need to be on its own subnet, and/or the degree of isolation from the rest of the network could be specified).

5.4.1.2 Operational Behaviour

5.4.1.2.1 Management Operations

Certain VNF lifecycle operations and events can be automated. The description of the VNF may include scripts linked to VNF lifecycle operations and events. These scripts can be used to automatically orchestrate (e.g. instantiation, monitoring and self-healing of VNF and all of its sub-components at run-time).

6 VNF States and Transitions

6.1 States and Transitions as Architectural Patterns

A VNF can assume a number of internal states to represent the status of the VNF. Transitions between these states provide architectural patterns for some expected VNF functionality. Before a VNF can start its lifecycle, it is a prerequisite that the VNF was on-boarded (process of registering the VNF with the NFVO and uploading the VNF data (VNFD, SW images etc). On-boarding is the responsibility of NFVO and therefore not described here.

A list of states that a VNF can assume is presented in table 2.

Table 2: VNF Instance States

State	Description
Null	A VNF Instance does not exist and is about to be created.
Instantiated Not Configured	VNF Instance does exist but is not configured for service.
Instantiated Configured - Inactive	A VNF Instance is configured for service.
Instantiated Configured - Active	A VNF Instance that participates in service.
Terminated	A VNF Instance has ceased to exist.

Table 3 describes the state transitions in details.

Table 3: VNF state Transitions

Transitions action	Reverse transition action	Description
Instantiate	Terminate	Instantiate results in the creation of a VNF instance in an Instantiated Not Configured state. The NFVI resource(s) necessary for a particular VNF instance are allocated and in use. Terminate deletes a previously created VNF instance. All NVFI resources are released and made available for other VNFs.
Configure	N/A	Configure sets or changes VNF instance's parameters in such way that it enables the VNF instance to participate in a desired service, e.g. virtual DRA (vDRA) configured for participation in VoLTE.
Start	Stop	Start results in an Instantiated Configured - Inactive VNF Instance changing to an Instantiated Configured - Active state that triggers it to perform its functionality (e.g. able to accept messages from external Network Functions and sending responses). Stop results in a n Instantiated Configured - Active VNF Instance starting a "shut-down". This may involve multiple steps, such as terminating jobs and processes or releasing allocated memory. After the operation is fully executed, a VNF Instance is in an Instantiated Configured - Inactive state, but ready to be started.
Scale out	Scale in	Scale out results in adding additional VNFC instance(s) to a VNF instance. New VNFC instance(s) will require new virtualisation containers with compute, network, and/or storage capacity. Scale in will remove VNFC instance(s) and their VMs, freeing up compute, network and storage that are resources no longer needed. Scale in and out can be multidimensional, i.e. it would be possible to scale in/out VNF instance by removing/adding a VNFC instance(s) associated with throughput in VNF and/or other VNFC instance(s) associated with sessions or signalling.
Scale-up	Scale-down	Scale up results in VNFC instance(s) increasing the compute, network, and/or storage resources that it/they use(s) (e.g. replace a dual-core with a quad-core processor). Scale down results in VNFC instance(s) decreasing compute, network, and/or storage resources that it/they use(s) (e.g. replace a quad-core processor with a dual-core processor).

Transitions action	Reverse transition action	Description
Update	Update Rollback	Update results in the deployment of fixes that do not create new functionality. This includes corrections to existing functionality (e.g. bug fixes) as well as improvements to existing functionality. Update Rollback results in removing the changes (fixes and/or improvements) provided by the update.
Upgrade	Upgrade Rollback	Upgrade results in the deployment of an enhanced version of a VNF. After the operation is fully executed, a VNF instance is based on a new version. Upgrade Rollback results in returning the functionality of a VNF to a previous version of VNF.
Reset	N/A	Reset sets a VNF Instance that is currently in Instantiated Configured (Inactive, Active,) into Instantiated Not Configured state.

Figure 20 provides a graphical overview of the VNF states and state transitions.

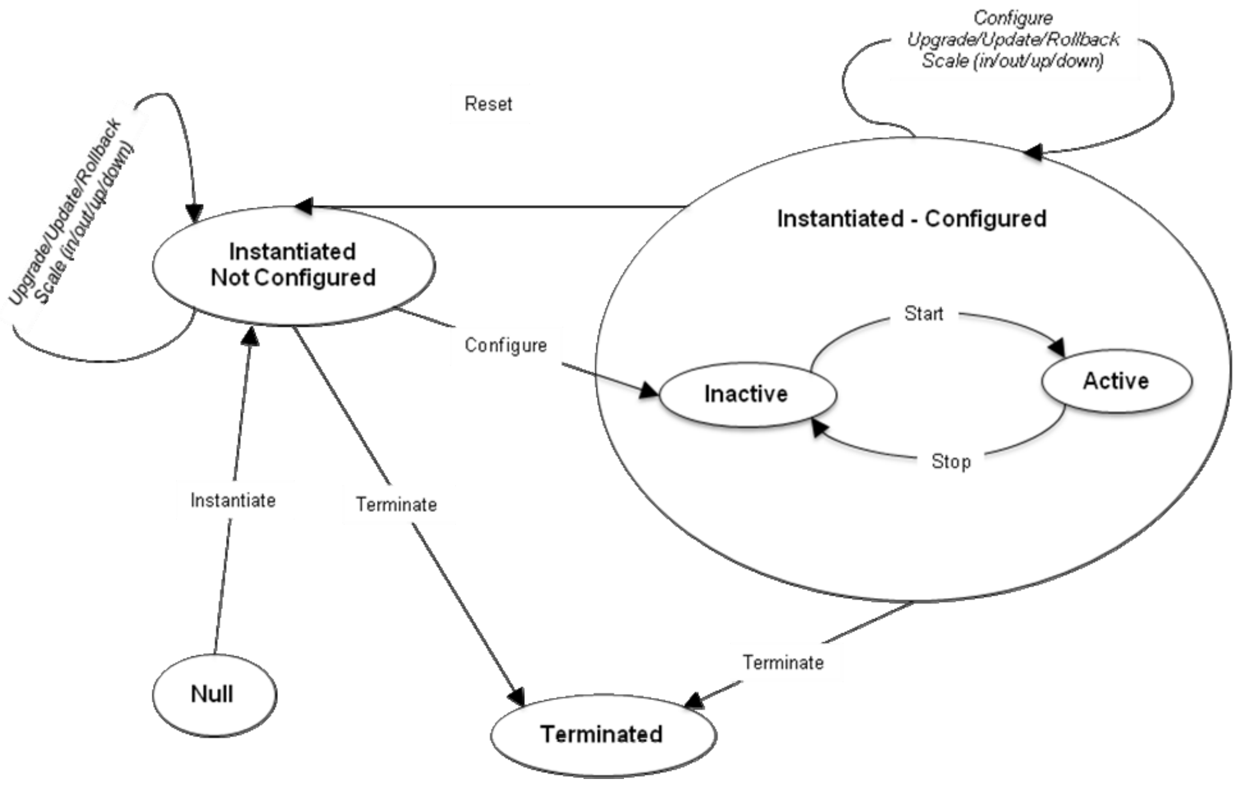

Figure 20: VNF instance state transitions

6.2 VNF Instantiation Overview

Depending of the type VNF the instantiation will be more or less complex. For a simple VNF consisting of a single VNFC the instantiation based on VNFD is straightforward, while for a complex VNF consisting of several different VNFCs connected via virtual networks the VNFM may require support from a internal functionality that VNF implements to facilitate the process of instantiation.

As an example, the VNFM could boot a pre-defined number of VNF Components, leaving booting of the remaining VNFCs and the configuration of the VNF to an internal VNF Provider specific process that may also involve an EM. A simplified example of the instantiation case is shown in figure 21.

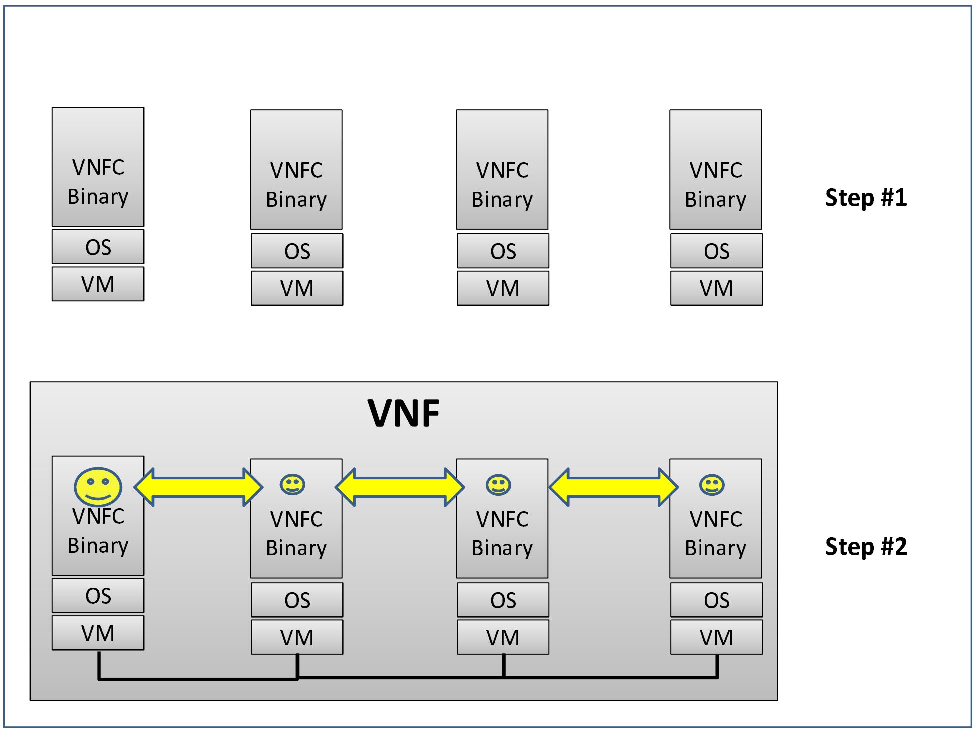

Figure 21: VNF Instantiation Steps

A hypothetical VNF is made up of 4 VNFCs. All VMs used by each VNFC instance are located on the same VLAN. In step #1 of the instantiation, the VNFM creates four complete VNFC instances. At this point, there is no VNF instance; rather, there are four separate VNFC instances that at best trying to interconnect. The VNF, consisting of four interconnected VNFC instances, is created during step #2 of the instantiation. In this example, each VNFC instance uses a function that broadcasts messages to other VNFC instances looking for a VNFC instance that implements a master function that will organize the VNFC instances, coordinate their actions, and make them act as one functional unit. The "big smiley" in figure 21 represents the master function. The master function uses the Ve-Vnfm-vnf interface to interact with the VNF Manager.

Layer 2 and/or Layer 3 addresses need to be allocated to VNF and VNFC instances to enable communication with other instances, the EM and the VNFM. Such addresses can be allocated to the VNF at instantiation time (i.e. addresses are included in the instantiation request) and/or during or after instantiation (e.g. scale-out) by the NFV management and orchestration functions upon requests embedded in VNF lifecycle workflow scripts (see clause 5.4.1.2.1). VNF lifecycle workflows scripts can also determine which VNFC instances within a VNF instance are bound to these addresses and when.

6.3 The VNF Descriptor's Role in VNF Instantiation

The VNFD is a specification template provided by the VNF Provider for describing virtual resource requirements of a VNF. It is used by the NFV Management and Orchestration functions to determine how to execute VNF lifecycle operations (such as instantiation) [i.3].

Figure 22 illustrates a simplified example of a VNF Descriptor (VNFD) and its relationship with the VNF.

The example in figure 22 shows a VNF instance that is made up of 4 VNFC instances, which are of 3 different types: 'A', 'B' and 'C'. Each VNFC type has its own requirements on the operating system (OS) and the execution environment (e.g. the virtual machine). These virtual resources and their interconnectivity requirements are described in data model elements that form a VNFD. Besides resource requirements, a VNFD also contains unambiguous references to VNF binaries, scripts, configuration data, etc., that are necessary for the NFV management and orchestration functions to configure the VNF properly.

The requirements for the NFVI resources (e.g. connectivity requirements, bandwidth, latency, etc.) are not shown in figure 22, but are assumed to be present in the VNFD as well as in other descriptors used by the NFV management and orchestration functions. The VNFD can also specify location rules (e.g. some VNFC instances need to be on the VMs provided on the resources located in the same rack).

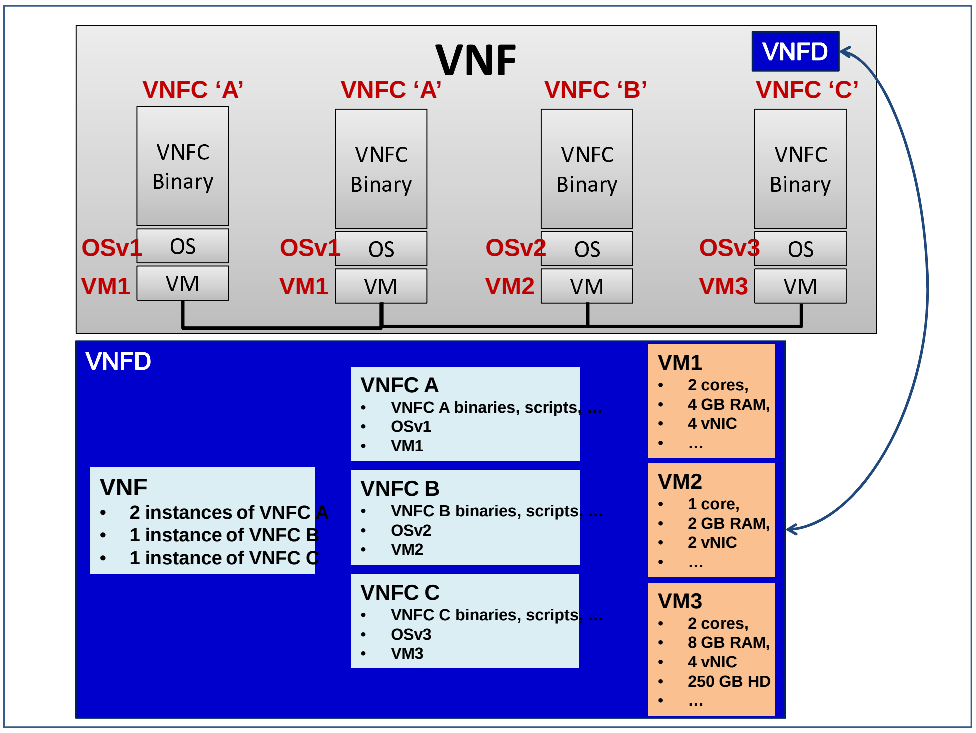

Figure 22: Example of VNFD Elements

The NFV management and orchestration functions take into available account all VNFD attributes to check the feasibility of instantiating a given VNF, e.g. check what type of resources each VNFC instance requires. There are several options for how the instances of individual VNFCs can be created (refer to listing above), namely:

- Fully or partly loaded virtualisation containers; or
- Empty virtualisation containers prepared for booting and loading.

For example, if the latter option is used, then the virtualisation container loading file is NULL. It is then the responsibility of the VNF management and orchestration functions (i.e. NFVO and VNF Manager) to instruct the VIM to create an empty virtualisation container with an associated SWA-5 (Vn-Nf) interface that is ready for use. The specifics on how VNF/VNFC are created are outside the scope of the present document.

6.4 VNF Instantiation

Pre-conditions:

- The VNF has been on-boarded.
- The VNF's VNFM is in service.
- If the VNF uses an external EM, that EM is in service.

To instantiate a VNF, the VNF Manager creates the VNF's set of VNFC Instances as defined in the VNF Descriptor by executing one or more VNFC Instantiation procedures (see clause 6.5). Which VNFC Instances may be created in parallel and which shall be created sequentially as well as the order of instantiation is defined in the VNF Descriptor.

The set of created VNFC Instances may already correspond to the complete VNF Instance. Alternatively, it may contain only a minimal set of VNFC Instances needed to boot the VNF Instance. The boot VNF Instance (or its EM, respectively) may then request the VNF Manager to perform further VNFC Instantiation procedures until the VNF Instance is complete. For this, the VNF may, for example, contain a boot server in one of the early VNFC instances used to boot up the remaining VMs. The VNFC instantiation procedure is described in clause 6.5.

The VNF Instance may at any time during the instantiation process register with its EM (if applicable, and a virtual network is set up) and receive missing configuration information, e.g. OSS-provided policies, according to the VNF Provider's implementation. At this point, the VNF Instance is instantiated but not configured, and the VNF Instance informs its VNF Manager.

Post-conditions:

- The VNF Instance is in Instantiated NOT Configured state.

6.5 VNFC Instantiation

Pre-conditions:

- The VNF has been on-boarded.
- The VNF's VNFM is in service.
- If the VNF uses an external EM, that EM is in service.

The VNF Manager requests a new VM and the corresponding network and storage resources for the VNFC Instance according to the definition in the VNF Descriptor or uses a VM and the corresponding network and storage resources previously allocated to it. After successful completion, the VNF Manager requests to start the VM.

After boot-up of the VM and depending on the information of the VNF Descriptor, the VNF Manager:

- may inject basic instance configuration into the VNFC and start additional processes on the VM according to the specification in the VNF Descriptor;
- shall inform the VNF or its EM that the VM is ready for configuration.

The running VNFC Instance may communicate to the virtual networks or virtual storage connected to the VM. It may use other VNFC Instances and/or the VNF's EM to receive further configuration according to the VNF Provider's implementation. This may include reconfiguration of internal load balancers.

The VNF Manager and EM is informed when the VNFC Instance is fully created, configured and ready for operation.

Post-conditions:

- The VNFC Instance is instantiated.

6.6 VNFC Instance Termination

Pre-conditions:

- The VNFC Instance is operational.

The VNF Manager requests to shut down the VNFC Instance, allowing, e.g. for reconfiguration of load balancers and transfer or saving of VNFC Instance state.

Once the VNFC is gracefully shut down, VNF informs the VNFM that takes appropriate actions.

A VNF, when it has appropriate functionality, may itself initiate termination of its VNFC Instance. In such case, the VNF sends a request to its VNFM that needs to be granted before the VNF takes further action.

Post-conditions:

- The virtual resources used by the VNFC Instance can be deleted.

6.7 VNF Instance Termination

Pre-conditions:

- The VNF Instance is in Instantiated Not Configured or Instantiated Configured (Inactive, Active) state.

To terminate a VNF Instance, the VNF Manager may do one of the two actions:

1) Terminate the VNFC Instances as defined in the VNF Descriptor by executing one or more VNFC Instance Termination procedures. Which VNFC Instances may be terminated in parallel and which shall be terminated sequentially as well as the order of termination is defined in the VNF Descriptor.

2) Send the terminate request to the VNF that executes an internal procedure that gracefully shuts down the VNF, e.g. adjusting load balancers and transferring of saving state. Once the VNF is not performing any new tasks and not accepting new tasks, it informs VNFM that it is ready for termination. At that point, VNFM executes the relevant procedure for VIM to release VNF resources.

If a graceful shutdown fails, the VNF Manager may request to forcefully shutdown all of the VNF Instance's VMs and allow VIM to release the associated resources.

Post-conditions:

- The VNF Instance ceased to exist and all resources can be released.

6.8 VNF Instance Scaling

6.8.1 General Aspects

A VNF is scaled by scaling one or more of its constituent VNFCs, depending on the type of work load. For example, a VNF may be scaled for its user plane VNFCs, its control plane VNFCs or both.

Scale out/in is implemented by adding/removing VNFC instance(s) that belong to the VNF instance being scaled.

Scale up/down is implemented by changing (adding/removing) resources from existing VNFC instance(s) that belong to the VNF instance being scaled.

6.8.2 Scaling Triggers

Clause 5.1.5 defines the three principal mechanisms for triggering the scaling of a VNF Instance:

1) Auto-scaling, in which the VNF Manager monitors the state of a VNF Instance's constituent VNFC Instances and triggers the scaling when certain conditions are met. For monitoring a VNF Instance's state, it can for instance track infrastructure-level and/or VNF-level events. Infrastructure-level events are generated by the VIM. VNF-Level events may be generated by the VNF Instance or its EM.

2) On-demand scaling, in which a VNF Instance or its EM monitor the state of a VNF Instance's constituent VNFC Instances and trigger scaling through explicit request to the VNF Manager.

3) Scaling based on a management request, in which a higher-layer management system (e.g. via NFVO as an entry point for requests from OSS/BSS) or human operator triggers scaling through explicit request to the VNF Manager.

6.8.3 VNF Scale-out

Pre-conditions:

- The VNF Instance is in Instantiated Not Configured or Instantiated Configured (Inactive, Active) state.
- The VNF supports scale-out.
- If the VNF uses an external EM, that EM is in service.

The VNF Manager performs one or more VNFC Instantiation procedures.

The VNF Manager gives the VNF Instance or its EM a post-scale-out notification, so that it may perform potentially necessary internal operations after the scale-out.

In case of on demand scale-out, a VNF sends a request to its VNFM for scale out. If the request is granted, VNFM performs the VNFC Instantiation procedure.

Post-conditions:

- The VNF Instance is scaled out and its state is unchanged.

6.8.4 VNF Scale-in

Pre-conditions:

- The VNF Instance is in Instantiated Not Configured or Instantiated Configured (Inactive, Active) state.
- The VNF supports scale-in.

The VNF Manager performs one or more VNFC Instance Termination procedures.

The VNF Manager gives the VNF Instance or its EM a post-scale-in notification, so that it may perform potentially necessary internal operations after the scale-in.

In case of on demand scale-in, a VNF sends a request to its VNFM for scale in. If the request is granted, the VNFM performs the VNFC Termination procedure.

Post-conditions:

- The VNF Instance is scaled in and its state is unchanged.

6.8.5 VNF Scale-up

Pre-conditions:

- The VNF Instance is in Instantiated Not Configured or Instantiated Configured (Inactive, Active) state.
- The VNF supports scale-up.

The VNF Manager requests an update of the resources of one or more of the VNFC Instance(s) to be scaled-up, based on the received scaling request (if not auto-scaling) and constrained by the VNFD.

The VNF Manager gives the VNF Instance or its EM a post-scale-up notification, so that it may perform potentially necessary internal operations after the scale-up.

Post-conditions:

- The VNF Instance is scaled up and it state is unchanged.

6.8.6 VNF Scale-down

Pre-conditions:

- The VNF Instance is in Instantiated Not Configured or Instantiated Configured (Inactive, Active) state.
- The VNF supports scale-down.

The VNF Manager requests an update of the resources of one or more of the VNFC Instance(s) to be scaled-down, based on the received scaling request (if not auto-scaling) and constrained by the VNFD.

The VNF Manager gives the VNF Instance or its EM a post-scale-down notification, so that it may perform potentially necessary internal operations after the scale-down.

Post-conditions:

- The VNF Instance is scaled down and it state is unchanged.

6.9 Start and Stop VNF

6.9.1 Start VNF

Pre-conditions:

- The VNF Instance is in Instantiated Configure-Inactive state.

The VNF Manager or the EM requests the VNF Instance to transit to an active state where it is ready to perform its functionality and (e.g. able to accept messages from external Network Functions and send responses).

The VNF Instance informs its VNF Manager about the start.

Post-conditions:

- The VNF Instance is in Instantiated Configured-Active state.

6.9.2 Stop VNF

Pre-conditions:

- The VNF Instance is in Instantiated Configured-Active state.

The VNF Manager or the EM requests the VNF Instance to start a "shut-down".

The VNF Instance informs its VNF Manager about the stop.

Post-conditions:

- The VNF Instance is in Instantiated Configure-Inactive state.

6.10 VNF Instance Configuration

Pre-conditions:

- The VNF instance is in Instantiated Not Configured state or in Instantiated Configured (Inactive, Active) state.
- The VNF's VNFM is in service.
- If the VNF uses an external EM, that EM is in service.

A VNF Instance can be configured by its VNF Manager and/or its EM. The entity performing the configuration is referred to here as the configurator.

After the VNF instance has been configured, it informs the configurator.

Post-conditions:

- The VNF Instance is in Instantiated Configured (Inactive, Active) state.

7 VNF Fault Management Overview

7.1 Introduction

One of the principles of the NFV framework is to mitigate hardware failures (from the decoupled VNF software) by masking NFVI (hypervisor and hardware) failures, and by leveraging virtualisation-based high-availability mechanisms, e.g. virtualisation container migration. By utilising such techniques, in general, VNFs become independent of NFVI faults. However, for the lifecycle management of a VNF, general NFV fault management principles and requirements are still relevant.

Fault management includes several processes including:

- Fault detection.
- Fault localisation.
- Fault reporting.

These processes can be located and performed by different entities. Therefore VNF fault management handling needs to work jointly with several functional elements in the NFV Architectural Framework that are responsible for the virtualised resource health-state monitoring and data collection.

From a high-level perspective, a VNF needs to be informed about faults in the entities it depends on for its lifecycle, i.e. virtualised resources. Along with that, a VNF needs to inform management entities responsible for its lifecycle management about faults occurring in the VNF itself. As a result, VNF fault management encompasses:

1) Faults involving the virtualised resources allocated and consumed by a VNF.
2) Faults within a VNF.

7.2 Virtualised resource faults

When the network function is virtualised, typical hardware faults are handled by NFV infrastructure, and management and orchestration functions, and only information about those faults (e.g. fault corresponding to virtualised resources) that affect the VNF's proper functioning need to be provided to a VNF.

Moreover, all hardware faults that are generated by the NFVI are on first instance detected by the own NFVI elements and corrected jointly with the support from the NFV management and orchestration functions (e.g. VIM). The management and orchestration functions, are then also responsible for providing virtualised resource fault information to corresponding VNF fault management functions (e.g. EM).

The following list shows items and faults from the underlying NFVI which can have impact on a VNF:

- Fault in virtualised resources that might affect a VNF's proper functioning (e.g. whole NFVI down).
- Fault in the VNF's redundancy scheme (e.g. backup virtualised resources unavailable).
- Fault in the Vn-Nf/SWA-5 interface (e.g. fault in the virtualisation layer/hypervisor).
- Fault in the virtualisation container (e.g. VM malfunctioning).
- Faults concerning virtualisation container connectivity.
- Others.

Fault information on the above needs to be sent by responsible functional elements using virtualisation resource fault management interfaces, as defined in [i.3] so that the VNF and/or corresponding fault management functions (e.g. within EM) can take appropriate actions.

7.3 VNF faults

VNF faults concern faults within the VNF application itself. This can include faults such as:

- Software bug related faults, e.g. code exceptions.
- Communication failures among VNFCs.
- Security faults (e.g. VNF security breaches, denial of service, etc.).
- VNF configuration failures (e.g. wrong configuration of service parameters, error loading a configuration file, etc.).
- Others.

Fault information on the above need to be sent from the VNF to the corresponding functional elements responsible for taking necessary actions (e.g. EM, VNFM) using the VNF fault management interfaces defined for such a purpose [i.3].

8 Functional Requirements on Management and Orchestration

8.1 High Level Requirements to Management and Orchestration

8.1.1 General Management and Orchestration Requirements related to VNF

REQ number	Description
REQ-001	There are different types of VNF and network service structures, with different management, deployment and operational needs, as well as with different management models. There shall be a common and standard mechanism for VNF Providers to describe their VNFs and VNFCs, in terms of deployment and operational model, and the functions relying on a management and orchestration system over open standard interfaces, e.g. scaling, fault recovery, etc. (see note 1).
REQ-002	A common standard VNF Descriptor format shall be used by the NFV management and orchestration in order to be able to deploy a network service built with VNFs from multiple VNF Providers.
REQ-003	A VNF may support a redundancy model (e.g. active/standby). The service logic to handle redundancy may reside in different functional elements depending on the VNF Provider implementation as well as operator's operational policy (EM, VNF Manager, VNF, OSS, or NFV Orchestrator).
REQ-004	NFV management and orchestration shall be able to execute processes required to support VNF state transitions, such as: provision/deployment of a VNF, configuration of a VNF, starting and stopping the execution of a VNF instance and creation/termination of a VNF instance. Configuration of the VNF may include configuration of the virtualised environment of the VNF (e.g. IP@, networking aspects, load balancer etc.), and/or configuration of some logical functions of the VNF. This configuration information will be used either during instantiation of the VNF or after instantiation.
REQ-005	NFV management and orchestration shall support host affinity and anti-affinity rules for deployment of VNFC instances of a VNF on the same or separate physical hosts. These rules shall also be applied at later VNF state transitions like migration and scale-out.
REQ-006	NFV management and orchestration shall support management and orchestration of several isolated virtual networks interconnecting the VNFC virtualisation containers of a VNF.
REQ-055	NFV management and orchestration shall be able to verify that the selection of NIC and CPU are compatible with the requested data processing acceleration library (see note 2).
NOTE 1: A VNFC is deployed as defined by its corresponding VDU. NOTE 2: Future work (in addition to REQ-055) is needed to determine requirements pertaining to accelerator libraries (e.g. requirements for proper interworking with NFV management and orchestration.)	

8.1.2 Management and Orchestration Requirements Related to VNF Lifecycle

REQ number	Description
REQ-007	The NFV management and orchestration VNF lifecycle operations shall support authorization of the lifecycle management request to ensure that such VNF lifecycle operations are consistent with service provider policies, the VNF Descriptor, and resource constraints in order to avoid network service interruption.
REQ-008	The NFV management and orchestration VNF lifecycle operations shall support validation of the lifecycle management request to ensure that such VNF lifecycle operations are consistent with service provider policies, the VNF Descriptor, and resource constraints in order to avoid network service interruption. For example, when a VNF instance within a VNF-FG is terminated by NFV management and orchestration but replaced by another VNF, the impact on service using that VNF should be checked and such impact should be within a pre-determined service availability range.
REQ-009	VNF lifecycle operation requests shall be limited to some "authorized" functional elements.
REQ-010	The VNF Descriptor shall provide the NFV Orchestrator and the VNF Manager with information for performing the lifecycle management of a VNF as part of the network service.
REQ-011	The VNFD shall allow for different processes in support of the VNF state transitions described in the present document. For example, the VNF specific data required for the state transition is the one that is specific for the VNF Manager and VNF Package.
REQ-012	The NFV Orchestrator shall be able to deploy the network services composed of VNFs with different processes in support of the VNF state transitions described in the present document.
REQ-013	The VNF Package shall provide sufficient information for performing the instantiation of the VNF.
REQ-014	An NFV Orchestrator from any vendor shall be able to interact with the VNF Manager of a VNF that comes from any other vendor.
REQ-015	The VNF Package shall be provided in a format that can be stored in corresponding management and orchestration repositories.
REQ-016	A VNF shall have at least one SWA-1 interface. A VNF may support multiple SWA-1 interfaces (see note 1).
REQ-017	A VNF SWA-1 interface shall be individually identifiable.
REQ-018	A VNF shall have at least one SWA-5 interface. A VNF may support multiple SWA-5 interfaces.
REQ-019	A VNF SWA-5 interface shall be individually identifiable.
REQ-019bis	It shall be possible to allocate a set of addresses to a VNF at instantiation time (see note 2).
REQ-019ter	The NFV management and orchestration shall be able to dynamically allocate or deallocate addresses of a given type to a VNF instance and/or a VNFC instance during life cycle management operations based on rules that can be specified by the VNF Provider.
REQ-019quarter	It shall be possible to assign to a VNF addresses of the following types: L2 addresses, IPv4 and IPv6 addresses, private and public IP addresses, multicast IP addresses, and anycast IP addresses.
NOTE 1: A VNF that does not have the capability to connect to any other VNF, PNF, or End Point cannot be part of a network service. NOTE 2: Allocation of these addresses to VNFC instances can be driven by lifecycle management operations based on rules that can be specified by the VNF Provider.	

8.1.3 Management and Orchestration Requirements Related to Scaling

REQ number	Description
REQ-020	Scaling event types and format shall be defined.
REQ-021	Upon request, the NFV management and orchestration shall be able to change the amount of resources allocated to each VNFC and/or to change the number of instances of each VNFC for scaling or redundancy.

8.1.4 Management and Orchestration Requirements Related to VNF Maintenance Tasks

REQ number	Description
REQ-022	A VNF shall support upgrades, which consist of deploying one or more new SW image(s).
REQ-023	The NFV management and orchestration of VNF update/upgrade operation shall support validation of the VNF update/upgrade management request to ensure that such VNF update/upgrade operations are consistent with service provider policies, the VNF Descriptor, and resource constraints to avoid network service interruption.
REQ-024	The NFV management and orchestration shall maintain records on VNF Packages (e.g. SW image catalog, VNFD, version info) involved in a VNF update/upgrade operation.
REQ-025	A VNF may support upgrades that consist of deploying a new version, which may involve more than just deploying one or more new image versions (e.g. in order for the upgrade to operate correctly, it may require a set of different requirements, i.e. more memory). In this case the VNFD shall also be upgraded.
REQ-026	A VNF may support upgrades that require new functionalities from the underlying infrastructure (e.g. more hardware and/or virtual resources, or SRIOV driver). In this case the VNFD shall also be upgraded.
REQ-027	In case of change in the VNF lifecycle (e.g. termination of a VNF or a VNFC), NFV management and orchestration shall enable reporting of events back to VNF instances and/or interested clients (e.g. EM) due to changes made in the VNF components or used virtualised resources.

8.2 Requirements for VNFD and VNF-FGD Template

8.2.1 General Requirements Related to VNF

REQ number	Description
REQ-028	NFV management and orchestration functions may provide NFVI resources to a requested high availability (HA) VNF. If such NFVI resources are present, they may be directed to be placed in different physical hardware hosts and/or in different locations. It shall be possible to specify the requirements on specific NFVI resources as well as virtual, or geographically redundant, deployment in the VNFD.
REQ-029	When a VNF supports some form of redundancy, e.g. Active/Standby, then both of the deployment steps (i.e. deployment of the active instance, then deployment of the standby instance, configuration of the network, etc.) as well as the type of redundancy (i.e. Active/Standby - automatic detection and IP@ transfer) shall be specified. It shall be possible to describe the resources requested for Active and Standby and the redundancy models used in the VNF Descriptor (VNFD).
REQ-030	It shall be possible to specify the minimum and maximum number of VNFC instances required for deployment and elasticity of a VNF in the VNFD.
REQ-031	The VNFD shall describe the NFVI virtual resources required to deploy a VNF, such as the number of virtualisation containers, vCPU, and virtual memory per virtualisation containers.
REQ-032	When a VNF is made of multiple VNF Components, this requires information about the relationship (dependencies, inter-connectivity, etc.) between the VNFCs. It shall be possible to specify these properties in the VNFD.
REQ-033	Void.
REQ-034	The VNFD may have different versions to describe different VNF versions. Anytime there is an update on a VNF, the VNFD shall be updated and a new version of this VNFD shall be created.
REQ-035	A VNF requiring metrics to be collected for proper operation may indicate the need for batch delivery of metrics (e.g. round trip delay, jitter, etc.), providing a list of metrics required and eventually the source providing those (i.e. NFVI, other VNF, etc.), a set of conditions applying to those metrics (e.g. alarms=critical, or CPU > 70 %) and time range (e.g. last 3 days). It shall be possible to define this information in the VNFD, and/or ad-hoc request could be sent via the open interface with NFV management and orchestration (see note).
REQ-036	The VNFD of a given VNF shall provide sufficient information about the SWA-1 supported by that VNF to determine compatibility with a different VNF obtained from a different VNF Provider, such as certain protocol support.
REQ-037	The VNFD shall provide a parameter to specify that a VNFC requires a NIC compatible with a requested data processing acceleration software library.
REQ-038	The VNFD shall provide a parameter to specify that a VNFC requires a CPU compatible with a requested data processing acceleration software library (e.g. DPDK).
REQ-056	It shall be possible to specify in the VNFD which network speed/bandwidth is to be guaranteed per requested NIC.
REQ-057	It shall be possible to specify in the VNFD that a VNFC needs a NIC with RDMA support.
REQ-058	It shall be possible to specify in the VNFD that the VNFC needs a NIC that supports SRIOV.
NOTE: Implementations are expected to have VNFDs and VNF Managers that allow performance information sent to external EM, virtualised EM, the VNF Manager or the VNF instances.	

8.2.2 General Requirements Related to VNF Forwarding Graphs

REQ number	Description
REQ-039	A VNF Forwarding Graph shall have an associated VNF Forwarding Graph Descriptor (VNF-FGD), which defines metadata for the VNF Forwarding Graph (e.g. connectivity constraints).
REQ-040	The VNF-FGD shall reference to the VNF descriptors applicable to the VNFs requested, how many instances of each and any specific attributes and behaviours for each (e.g. active/standby deployment).
REQ-041	A VNF Forwarding Graph will describe the links between VNFs that are part of the VNF Forwarding Graph as well as links to PNFs this graph is connected to. It shall be possible to specify these properties in the VNF-FGD.
REQ-042	A VNF Forwarding Graph may require specific performance characteristics on the virtual network for inter-VNF communication (e.g. round trip delay, jitter, packet loss less than x %, etc.), as well as communication with external entities and PNFs (e.g. round trip delay, jitter, packet loss less than x %, etc.). It shall be possible to specify these requirements in the VNF-FGD.
REQ-043	A VNF Network Connectivity Topology (NCT) describes how one or more VNFs are connected to each other, regardless of the location and placement of the underlying physical network elements. A VNF NCT thus defines a logical network level topology of the VNFs in a graph, including connectivity between VNFs in the NCT and different NFs outside of the graph. The properties, relationships, and other metadata of the NCT shall be specified in Virtual Link Descriptors.
REQ-044	It shall be possible to asscociate multiple network forwarding paths to a forwarding graph, each of which associating a set of conditions to a sequence of connection points to be traversed by packets or frames matching these conditions. It shall be possible to describe the properties of network forwarding paths in the forwarding graph descriptor.

8.2.3 Requirements Related to VNF Creation and Termination

REQ number	Description
REQ-045	When a VNF is made up of multiple VNF Components, the VNF shall be maintained as one logical entity with one VNF Descriptor, and the NFV management and orchestration shall manage the VNF as one logical entity with identifier, descriptor, status, version, etc.
REQ-046	When a VNF is made up of multiple VNF Components, the VNF Descriptor shall include information to be used to deploy VNFC(s) contained in the VNF. NOTE: A VNFC is deployed by management and orchestration functions as defined by its corresponding VDU.
REQ-047	For a given VNF package, a VNF shall come with zero or one SW image per VNFC, and associated version number.
REQ-047bis	It shall be possible to support multiple SW images of the same VNFC by supporting multiple variants of a VNF Package. NOTE: Multiple SW images of the same VNFC may be required when a VNF is designed to execute on different types of virtualisation containers and/or different HW platforms.
REQ-048	The VNFD shall specify possible interfaces of such VNF with other entities, VNFs or PNFs. It will be up to the VNF-FG or NFV management and orchestration to determine or describe which interface(s) will be used by the other VNF, PNF entities.
REQ-049	The VNFD shall contain the specification of the connectivity graph and dependencies between the VNFC interfaces.
REQ-050	A VNF may be composed of multiple VNFCs that have different needs (e.g. number of vNICs, specific list of VLAN ids per vNIC, number of IP addresses per VLAN, bandwidth, unit cost, speed, security, etc.) and reliability/HA models (e.g. desired network resiliency, including e.g. link aggregation, NIC teaming, etc.). The VNFD shall support these types of VNFC requirements, including support for different redundancy models if requested.
REQ-051	The VNFD typically provided by the VNF Provider is not expected to be edited by any other entity. Management and orchestration shall ensure the integrity of this VNFD.
REQ-052	Some VNFs may require specific processing and storage capability, (i.e. dedicated CPU type or number of CPUs, dedicated memory, accelerators, etc.). It shall be possible to specify these properties in the VNFD.
Req-052bis	It shall be possible to specify in the VNFD whether to allocate/deallocate an address of a given address type to a VNFC instance when it is created/terminated.

8.2.4 Requirements Related to Scaling

REQ number	Description
REQ-053	A VNF supporting scalability will describe the different types of scalability supported (e.g. scale up, scale down, scale in, scale out) and describe how each one is achieved, for instance describing which VNFC(s) should be scaled. It shall be possible to specify these properties in the VNFD.
REQ-054	It shall be possible to specify in the VNFD whether to allocate/deallocate an address of a given address type to a VNFC instance when it is created/terminated as a result of scaling out/in.

9 Functional Requirements on Infrastructure

This clause captures the requirements VNF have on NFVI.

The SWA-5 interfaces are interfaces between the VNF and virtualisation containers. More formally, a virtualisation container provides interfaces for storing, querying, retrieving, managing, and executing objects that it contains. Hence, the virtualisation container defines the interface requirements for other objects to execute within its environment. Virtualisation containers can store objects directly, and/or be used to store references to objects. As such, the interface requirements are defined by the type of virtualisation container that the VNF interacts with. INF will define the requirements for this interface.

However, the VNF have certain requirements on the NFVI. Most of these requirements will translate into information exchanged with the Management and Orchestration, either via descriptors or interfaces.

Req #New_INF001: The NFV infrastructure shall support deployment of several isolated virtual networks interconnecting the VNFC virtualisation containers of a VNF.

Req #New_INF002: The NFV infrastructure shall support internal virtual networks with specified characteristics such as number of VLANS, VLAN ids, bandwidth, unit cost, speed, security, link aggregation, NIC teaming, etc.

9.1 Generic Domain Requirements

1) Generic Use of Roles:
    a) The use of roles is preferred to identifying specific entities, organizations, or services.
    b) or example, resource pools may be categorized by type, functional or non-functional requirements, or metadata (e.g. location, specific type of virtualisation used, etc.).
2) Generic Security Requirements:
    a) Roles should be used to abstract physical and virtual resources from how they are used.
    b) File access/Snapshoot access management and alarming to meet compliance requirements, and other security concerns - this needs to be part of both the Orchestrator, and the OS level views.
3) Generic Interoperability Requirements:
    a) Methodology to allow multiple Hypervisors from different VNF Providers to use the same template for an application.
4) Generic QoS/QoE Requirements:
    a) Define resource pools by type, where type is a role that can be mapped to an SLA, SLO, or SLS.
5) Generic Hardware Requirements:
    a) Hypervisor domain VM acceleration drivers for the OS application level (2 sets of drivers, one for the Hypervisor, the second for the OS) - Specialized HDW.
    b) NFVI nodes equipped with processors and NICs compatible with the selected data processing acceleration library.

6) Generic Performance Requirements:

   a) Hypervisor "CPU wait/ready time" readouts for application-level logging.

   b) OS-level drivers may be used to ensure smooth mouse, keyboard, and screen writes/reads for remote operations personnel to operate on that VNF.

7) Generic Virtual Data Centre (VDC) Requirements:

   a) A VDC is an abstraction that contains compute, storage, network, and virtualisation resources that:

      i) Supports multi-tenancy.

      ii) Enables role-based operations on their resources.

      iii) Maps to one or more physical resources.

   b) The VDC contains information about itself:

      i) One or more virtual networks and their properties (e.g. internal vs. external, whether they have firewall, NAT, or DPI functionality).

      ii) One or more VDC Catalogs (e.g. applications that can be used).

      iii) Specific performance, reliability, security, and other properties (e.g. HA, what type of redundancy).

      iv) Other (e.g. billing, AAA), which should be defined using roles.

   c) A VDC may contain one or more virtual resource pools.

8) Generic Capability exposure:

   a) Availability of software libraries and drivers shall be communicated to Management and Orchestration.

   b) Usage of software libraries and drivers shall be communicated to Management and Orchestration.

9.2 Hypervisor Requirements

The Hypervisor Domain is responsible for:

1) Abstracting the hardware to ensure portability of software appliances.

2) Allocating compute, network and storage domain resources to a given software appliance VM.

Use case:

1) Hypervisor VM to VM, VM to SAN, and I/O availability and activity monitoring are required to ensure that no problems have occurred (e.g. entities have not crashed or lost connectivity, or an entity to be restarted, or a cold or live migration is required).

This domain requires the following functionality:

1) Allocate the virtual resources needed by a VNF.

2) Add or remove virtual resources needed by a VNF for scalability.

3) Support affinity.

4) Support deployment of VMs that need access to bare metal (e.g. they are installed directly on the hardware, as opposed to within the host operating system).

5) Support deployment of VMs that need accelerators (e.g. DPDK, SRIOV).

6) Hypervisors with embedded vswitch or stand-alone vswitch shall support the selected data processing acceleration library.

9.3 Compute Resource Requirements

The Compute Domain is responsible for:

1) Abstracting compute functionality to ensure interoperability across VNF Providers.
2) Abstracting CPUs by multiple mechanisms, including (but not limited to) ISA and platform.
3) Providing management and orchestration interfaces.

Use case:

1) An OS-level driver of an application can trigger the OS to use its memory paging to offload the memory utilization of a hypervisor.

This domain requires the following functionality:

1) Provide the compute resources needed by the VNF.
2) Add or Remove resources assigned to a VNF to support scalability.

9.4 Network Resources Requirements

The Network Domain is responsible for:

1) Abstracting the network infrastructure to provide different types of connectivity services.
2) Providing multi-tenant address space and traffic isolation (if necessary).
3) Providing fast and flexible support services (e.g. DNS and DHCP services to support live migration).
4) Providing failover and protection services (e.g. LSP FRR; one-to-one or many-to-one LSP protection; per-node and/or per-link protection using bypass LSPs).

This domain requires the following functionality:

1) Simple, verifiable, virtual network isolation.
2) L2 edge services with Ethernet Addressing.
3) Ability to scale to a large number of virtual networks.
4) Attachment to both VNFs and PNFs.
5) Provide VNF IP address space isolation (when necessary).
6) Providing fast and flexible support services (e.g. DNS and DHCP services to support live migration, or load balancing, or DPI).
7) Providing failover and protection services (e.g. LSP FRR; one-to-one or many-to-one LSP protection; per-node and/or per-link protection using bypass LSPs).
8) The NFVI should be able to describe the characteristics of resources that it offers (e.g. the IP address, lease time if applicable, duplex and speed settings, etc.).
9) The NFVI should be able to describe specific endpoints that a VNF, or one of its subtypes, can communicate with (e.g. by using a DNS name, IP address, MAC address, or VSID).
10) The NFVI should be able to map the virtual network connectivity required by a VNF to the underlying physical infrastructure.
11) The NFVI should be able to describe how an individual or group of VNFs can access specific virtual network resource data (e.g. performance counters).

12) The NFVI should be able to describe specific endpoints that a VNF, or one of its subtypes, can communicate with; for example, a specific type of resource (e.g. a gateway to the Internet) may be requested, and the NFVI should be able to provide information about the requested resource (e.g. IP address of router or other device providing Internet connectivity).

13) The NFVI should be able to describe how an entity, such as a VNF, can obtain specific virtual network resource data.

14) The NFVI shall be capable of hosting servers with diverse capabilities (NIC supporting SRIOV, NIC supporting RDMA, Storage facilities with RDMA support, etc.).

15) The NFVI shall be capable of hosting servers with different network/NIC speed/bandwidth.

16) The NFVI shall expose to the VIM the capabilities supported by each server ((NIC supporting SRIOV, NIC supporting RDMA, Storage facilities with RDMA support, etc.) in order for the VIM to properly allocate resources to a VNF.

17) The NFVI shall expose to the VIM which network resources are available and their supported speed/bandwidth, in order for the VIM to properly allocate resources to a VNF.

18) The NFVI shall be able to commit/dedicate NIC speed/bandwidth to a VNFC.

10 VNF Architecture Design Examples

Network functions can be spread across data plane, and control plane primarily. Some functions as they are being virtualised can be deployed on a standard cloud infrastructure.

A VNF is realized by one or more VNFCs which appear as a single, integrated system in the network it supports. The details of the internal structure such as functional distribution between the VNFCs is not exposed, but internal connectivity between the virtualisation containers of the VNFCs needs to be specified by the VNF Provider, supported by the NFVI and managed by the NFV Management & Orchestration during the complete life-cycle of the VNF.

A number of network functions require specific resources or interactions: intense I/O, support real time services, are latency sensitive, need high data processing, use CPU memory data processing, memory data sharing, are coupled with other network functions which can be virtualised or not virtualised, have a need for high speed communication with other network functions, or with storage, etc.

All those requirements lead to architecture that enhance the current traditional cloud, with different technology such as DPDK, SRIOV etc and some software design that leverage all possible existing technology to provide the best cost/performance ratio. In some cases, specific functions might be offloaded to hardware to increase overall performance.

Assuming a given VNF is decomposed in VNFC, and that a VNFC maps 1-1 to a VM, the following clauses will describe different VNF design, interaction and usage models.

10.1 Faster VNFC

There are different ways to ensure a VNF will deliver performance [i.8]:

- Either by increasing the number of VNF or VNFC instances.
- Or by improving the design of the whole stack from the application on the VM to the lowest layers of the NFV-Infrastructure.

Figure 23 provides a few examples where optimisation is achieved using appropriate data processing acceleration libraries and drivers compatible with the CPU being used. This library can be called from either:

1) The network stack used by the application (A).

NOTE 1: It is up to the application developer to decide whether to develop a network stack that uses a data processing acceleration library or to rely on a 3rd party protocol stack that uses such a library.

2) A vswitch (B).

NOTE 2: The vswitch may be combined with the hypervisor as a single software package or provided as a standalone software package.

These examples assume that the acceleratin library is DPDK and the NFVI node is equipped with a Type I hypervisor and a vswitch.

NOTE 3: DPDK is an example of a suitable product available commercially. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of this product.

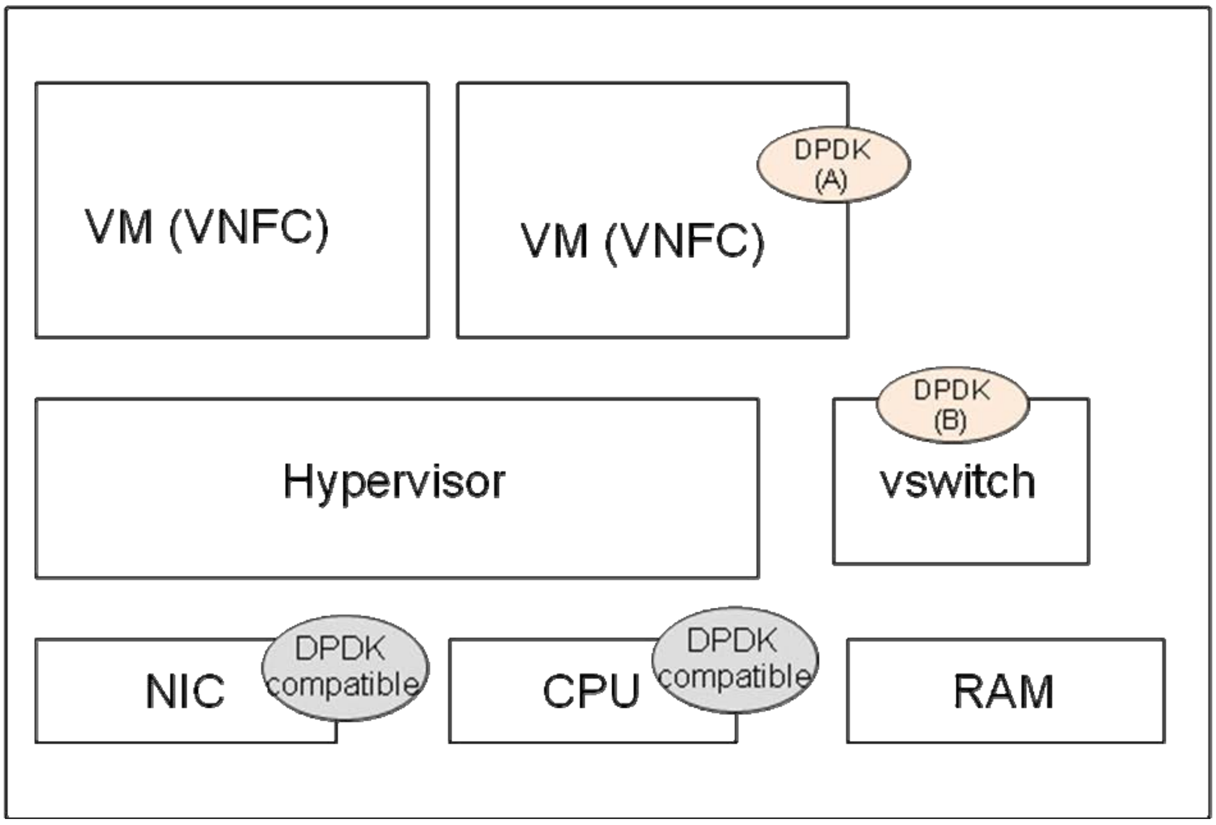

Figure 23: Faster VNF/VNFC

In all these examples, the processor needs to be compatible with DPDK and the NIC needs DPDK support in the form of a poll mode driver (DPDK typically comes with Poll Mode Driver for the NIC and DPDK libraries):

- Scenarios B do not require any change to the application compared to a PNF implementation.
- Scenarios A requires support of the DPDK library in the network stack.
- None of these scenario has impact on scalability or migration.

These scenarios could potentially be combined.

NOTE 4: DPDK, Data Plane Development Kit, is a set of open source libraries and drivers. More information is available on websites, such as www.dpdk.org.

10.2 VNFC to VNFC Communication

A VNF may need information to be passed from one VNFC to another within the same VNF, or across co-located VNFs. Depending on the application, e.g. voice/video, such inter-VNFC communications may have stringent QoS requirements.

Different network technologies can be used to support VNFC to VNFC communication (refer to figure 24):

1) Hardware switch (refer to scenario #1 in figure 24).

2) Virtual switch (vSwitch) deployed in the node hosting the VNFCs (refer to scenario#2 in figure 24).

3) vSwitch with data processing acceleration software deployed in the node hosting the VNFCs hypervisor (refer to scenario#3 in figure 24).

NOTE 1: The vSwitch may be combined with the hypervisor as a single software package or provided as a standalone software package running on top of or aside the hypervisor.

4) Embedded Switch (eSwitch) deployed in the NIC hardware with SR-IOV [i.19] (refer to scenario#4 in figure 24).

5) eSwitch deployed in the NIC hardware with SR-IOV, and with data plane acceleration software deployed in the VNFC (refer to scenario#5 in figure 24).

6) A Serial Bus (as described in clause 4.3.3) connects directly two VNFC that have extreme workloads or very low latency requirement.

NOTE 2: It complements shared memory scenario when the two VNFC cannot implement it (see clauses 4.3.3 and 10.3).

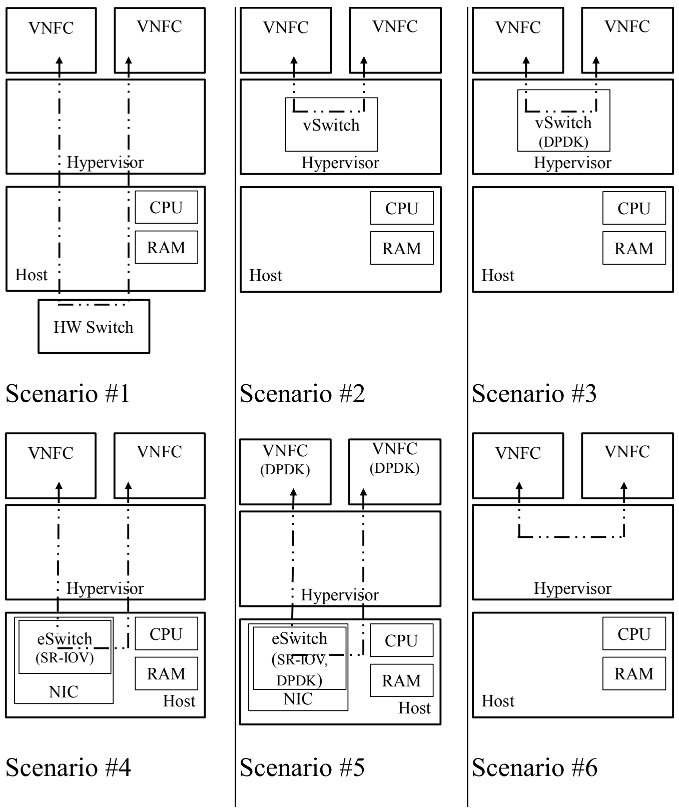

Figure 24: VNFC to VNFC Communication

NOTE 3: Similar techniques are also used to enable VNF to VNF connectivity [i.2].

NOTE 4: Figure 24 assumes that the data plane acceleration software is DPDK. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of this product. Equivalent products may be used if they can be shown to lead to the same results.

The above mentioned mechanisms derive certain deployment requirements:

- Scenarios 2 to 6 require VNFC affinity to support inter-VNFC communication within VNF.
- Scenario 4 requires the physical NIC to support SR-IOV.

- Scenario 5 requires the VNFCs to support data plane acceleration software, the physical NIC to support SR-IOV and be compatible with this software, and the processor to be compatible with this software as well.

10.3 VNFC Memory to VNFC Memory

Depending on the use cases, some network functions may require very low data path latency. For instance, in real time video conference, DPI may be on the data path but if video transcoding is required this needs to be performed very fast to avoid any latency and impact on the user experience or service quality. As mentioned in clause 4.3.3, VNFC memory to VNFC memory communication can serve such a purpose. In shared memory schemes, the shared part among VNFCs can be used for inter-VNFC communication purposes.

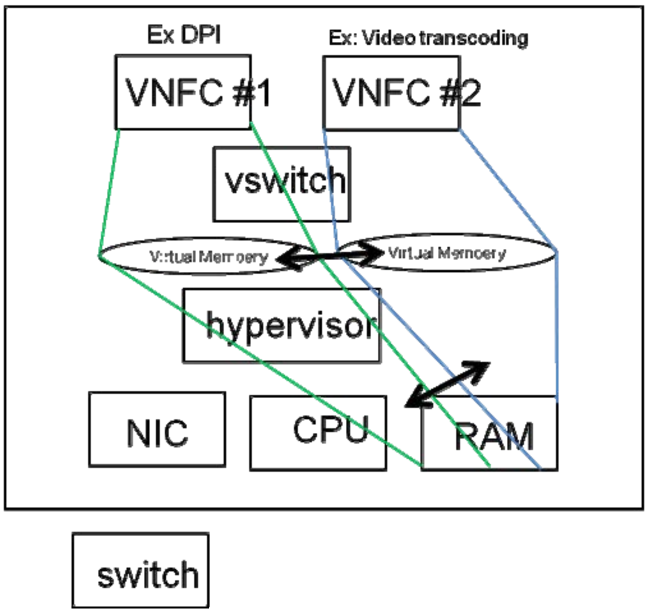

Figure 25: VNFC Memory to VNFC Memory

The abovementioned mechanisms derive certain deployment requirements (e.g. to NFV management and orchestration):

- This scenario requires VNFC affinity with support for shared memory (e.g. as in figure 25, VNFCs #1 & #2 need to be co-located on the same physical host).

10.4 Faster Network Access

Some Network Function, typically data plane functions, need to process high Gb Ethernet traffic. To ensure the latency between the NIC and the VM is optimized and high performance, technology like SR-IOV can be used.

The network speed/bandwidth is very relevant for the VM performance:

- ETH (Ethernet) support up to 40 Gbps today with a roadmap to 100 Gbps.
- IB (Infiniband) support up to 56 Gbps today with a roadmap to 104 Gbps:
    - IB networks are simpler to implement and manage and provide lower $/Gbps.

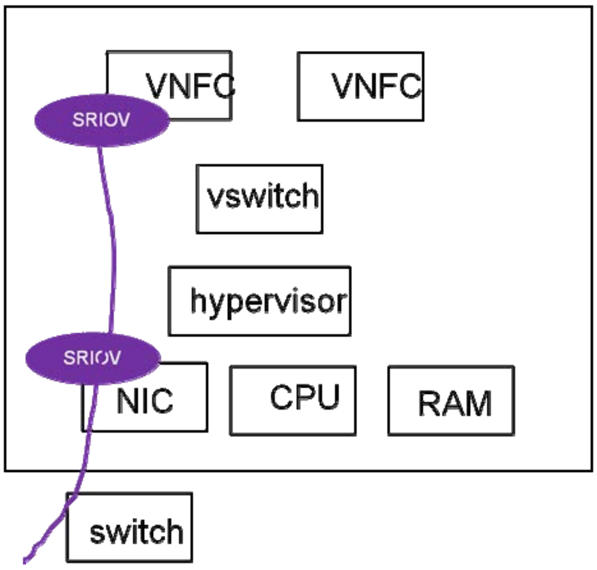

Figure 26: Fast Network Access

This enables achieving near bare metal performance, but requires that the VM using SR-IOV are deployed on servers with NIC supporting SR-IOV (industry standard).

10.5 Fast Storage Access

Some Network Functions deal with large amount of data, such as CDN and content, and use attached storage disk for instance with terabytes of data and fast access. For network functions where VNFC read and write data constantly to the storage server, communication from the compute server to the storage server should have the lowest possible latency and highest possible bandwidth, so all those CPU cycles are in fact processing, not waiting.

iSCSI predominates as a transport between storage and compute, and applications that use iSCSI over TCP are processed by the CPU, so the data path for iSCSI, as well as protocols such as TCP, UDP, and NFS, all have to wait in line with the other applications and system processes for their turn using the CPU. As the data channels cannot be locked down, it is hard to predict when the CPU or vCPU will have to give up processing power to get data through I/O. Add the layering of the virtualised environment, and the CPU is spending a lot of time working on managing how the OS handles this traffic. What is true for physical is also true for virtual CPUs.

Using RDMA (Remote Direct Memory Access) bypasses the operating system and CPU, allowing much more efficient data movement paths. iSER (iSCSI extensions for RDMA) capabilities are used to accelerate hypervisor traffic, including storage access, VM migration, and data and VM replication. The use of RDMA moves the data to the NIC, and provides zero-copy message transfers for SCSI packets, producing much faster performance, lower latency and access-time, and lower CPU overhead.

RDMA bypass allows data to be placed immediately upon receipt, without being subject to variable delays based on CPU load. This has three effects:

- There is no waiting, which means that the latency of transactions is incredibly low.
- Because there is no contention for resources, the latency is consistent.
- By skipping the OS, using RDMA results in a large savings of CPU cycles.

iSER can provide up to six times faster performance than traditional TCP/IP based iSCSI.

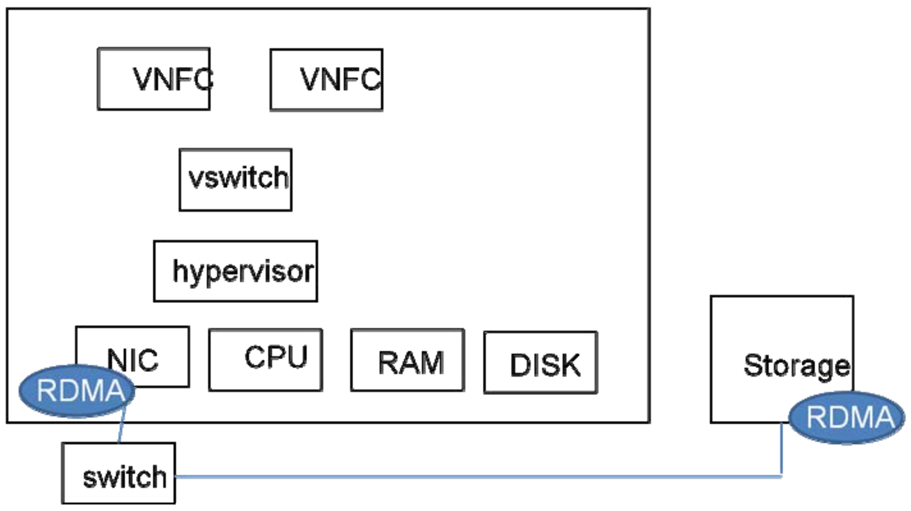

Figure 27: Fast Storage Access

The network speed/bandwidth is very relevant for the storage access and performance:

- ETH support up to 40 Gbps today with a roadmap to 100 Gbps.
- IB support up to 56 Gbps today with a roadmap to 104 Gbps:
  - IB networks are simpler to implement and manage and provide lower $/Gbps.

Fast and low latency storage access can be achieved by utilizing industry standard iSer protocol (RDMA/ROCE over iSCSI) over Ethernet or InfiniBand (IB) networks:

- RDMA: Remote direct memory access.
- RoCE: RDMA over Converged Ethernet.
- iSER (iSCSI extensions for RDMA).

NOTE: RDMA is open source provided by OpenFabrics Enterprise Distribution (OFED™) for kernel bypass applications https://www.openfabrics.org/index.php/ofed-for-linux-ofed-for-windows/ofed-overview.html.

10.6 Driver version, Software Updates

A number of the designs described earlier require some drivers. These drivers have versions and evolve. VNF Providers may want to update their driver versions, or VNF/VNFC may require specific driver version on a NIC for instance that will require a software update of this driver.

Requirements to Management and Orchestration:

1) VNFD should provide parameters that will allow VNF to specify driver version required for a given VNFC (i.e. SRIOV driver version on a NIC, etc.).

2) MANAGEMENT AND ORCHESTRATION should be able to instruct VIM to update a driver version if necessary, potentially passing the proper driver version to VIM for this update.

Requirements to VIM:

1) VIM should be able to describe which driver versions are being used.

2) VIM should be able to support driver version update.

NOTE: It is recommended to use a standard such as Openstack for driver management and updates (e.g. Quantum/Neutron agent on the NIC, Cinder agent on Storage etc) to have a common standard across the different VNF Provider platforms and across.

10.7 Distributed VNF

A VNF may be made up of one or more VNFCs, and each VNFC may be deployed in one or multiple instances. The VNF Provider may require that some VNFCs instances and their virtualisation containers are deployed on separate hosts in order to provide scalability and redundancy. The VNF may also require a set of internal networks interconnecting the virtualisation containers of the VNFC instances where the type of traffic carried on the different networks is internal to the VNF. The VNF could also provide connections to other VNFs or management functions but they are not described here.

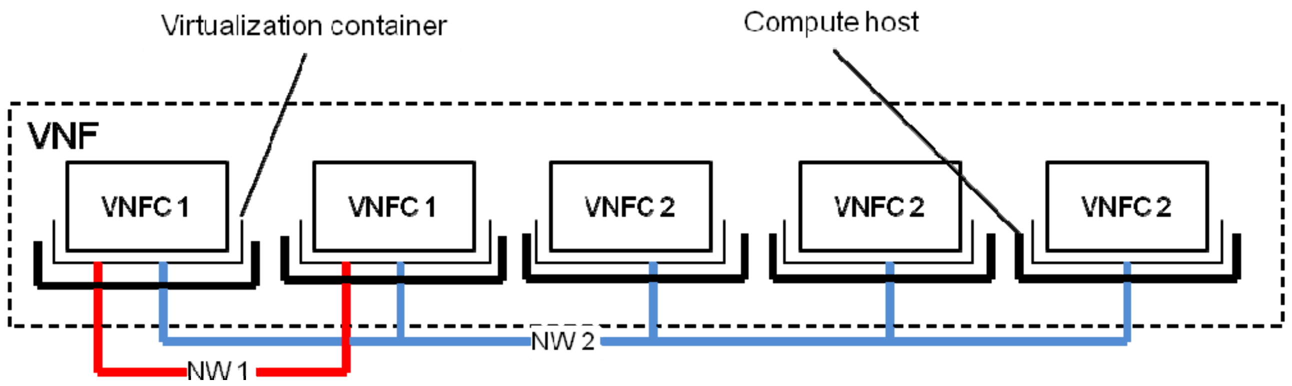

Figure 28: Distributed VNF

Figure 28 shows as an example a distributed VNF with two types of VNFCs (VNFC 1 and VNFC 2), 2+3 VNFC instances and two isolated logical networks interconnecting the virtualisation containers. Typically the VNF only requires layer 2 connectivity from the infrastructure while layer 3 and upwards is handled by means of VNF internals. Each network may consist of several VLANs.

The physical and virtual implementation of the networks, orchestrating vNICs, vSwitches, physical NICs and physical Ethernet switches, is made by MANO using the resources made available by one of the VIMs. The VNF is oblivious of actual physical (and virtual) network topology.

This leads to requirements on Management and Orchestration to support anti-affinity rules and multiple virtual networks and on NFV Infrastructure to support the virtual networks with characteristics specified by the VNF.

10.8 Generic VNFs

10.8.1 Definition and Usage

A generic VNF is a VNF general enough at its core functionality that it could be easily programmed for different usages yet keeping its core functionality intact. It has well-defined interfaces. Examples of such generic VNFs are load balancers, proxies, etc. In contrast, network functions like MME are specific in their functionality that they offer.

As an example, a load balancer can do load balancing in Layer 2~7. It can also perform complex load balancing on a protocol specific way or based on a combination of packet header information. A general usage can be found in conventional routing plane (refer to vCPE) and a specific usage that can be found in EPC (refer to annex C). However, its core functionality is load balancing itself and can be tuned for different usages when it publishes appropriate interfaces.

Annex A (informative): Relationship to SDN

A.1 Introduction to SDN

This clause first defines what is meant by SDN, and then introduces its main terminology and concepts. This is followed by a list of concerns for each SDN definition and architecture, and how that approach may impact NFV.

There are many different definitions of SDN. This clause provides definitions of SDN, and how they each relate to NFV.

A.1.1 ONF and SDN

ONF defines SDN as follows [i.18]: "The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices".

In other words, by pulling the control functions (or at least most of them) out of the network device, the network device may now be programmed by an external entity (an SDN Controller). ONF is focusing on OpenFlow as the protocol used between SDN Network elements and SDN controller.

A.1.2 OpenDaylight and SDN

OpenDaylight (www.opendaylight.org/) is an opensource project focused on SDN. OpenDaylight has defined SDN as "separates the control plane from the data plane within the network, allowing the intelligence and state of the network to be managed centrally while abstracting the complexity of the underlying physical network." (See http://www.opendaylight.org/project/technical-overview).

The main characteristics of OpenDaylight are:

- Support of Physical or virtual network devices, including OpenFlow enabled and open vswitch.
- Southbound support of multiple protocols incl OpenFlow, NetConf, SNMP, BGP-LS, etc.
- Controller with Service Abstraction Layer, Base Network Service Functions and specific services such as Affinity or Location and Virtual Tenant Network (VTN) which may be applicable to some aspects of Forwarding Graphs (See https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_(VTN):Main).
- Northbound REST API.

A.1.3 IETF and SDN

IETF has the following working groups that have defined or are in the processing of defining SDN-related protocols where control is separated from forwarding:

- FOrwarding and Control Element Separation (FORCES) http://tools.ietf.org/wg/forces/, which provides a feature-rich set of methods for a controller to configure, monitor and control physical or virtual forwarding elements.
- Interface 2 the Routing System (I2RS) http://tools.ietf.org/wg/i2rs/ is an effort to standardize the syntax for policy-based indirect control of routers.

Several specific IETF protocols are used in several controllers as a SouthBound Interfaces to control physical and virtual routers and switches:

- Border Gateway Protocol - Link State (BGP-LS) - http://tools.ietf.org/id/draft-ietf-idr-ls-distribution-04.txt is a set of extensions where the Traffic Engineering (TE) link state of an IGP can be exported up to a controller.
- Path Computation Element Protocol (PCEP) - http://tools.ietf.org/wg/pce/ supports a PCE as a either a stateless query or stateful direction of MPLS Traffic Engineered (TE) Label Switched Paths (LSP).

LISP (Location Identifier Separation Protocol) is an overlay architecture allowing the separation of Location from Identity, by allowing network forwarding element to query a Mapping Service for the current location (RLOC) of a particular end-point, specifically a destination IP.

Other IETF groups are doing SDN-related research and requirements, but are not mentioned here since there is no committed plan to develop a protocol and hence a gap analysis does not make sense at this point in time.

Other IETF groups that are doing work related to SWA, but not specifically related to SDN (e.g. NVO3, SFC, VNFPOOL, OPSAWG) are not covered in this SDN related annex.

A.1.4 ITU-T and SDN

A.1.4.1 Introduction

Recommendation ITU-T Y.3300 [i.33], Framework of Software-Defined Networking, defines SDN as "a set of techniques that enables to directly program, orchestrate, control and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner".

A.1.4.2 Protocol: SDN control

The Gateway Control Protocol (GCP) according to Recommendation ITU-T H.248.1 [i.20] (thus, also known as "H.248" protocol) is the basic solution for decomposed network elements for network functions "L2 switch", "packet gateway", "IP router", "media aware interworking functions", etc. H.248 is a candidate solution for the "southbound, vertical" control interface in SDNs. Figure A.1 highlights the high level models between OpenFlow and H.248.

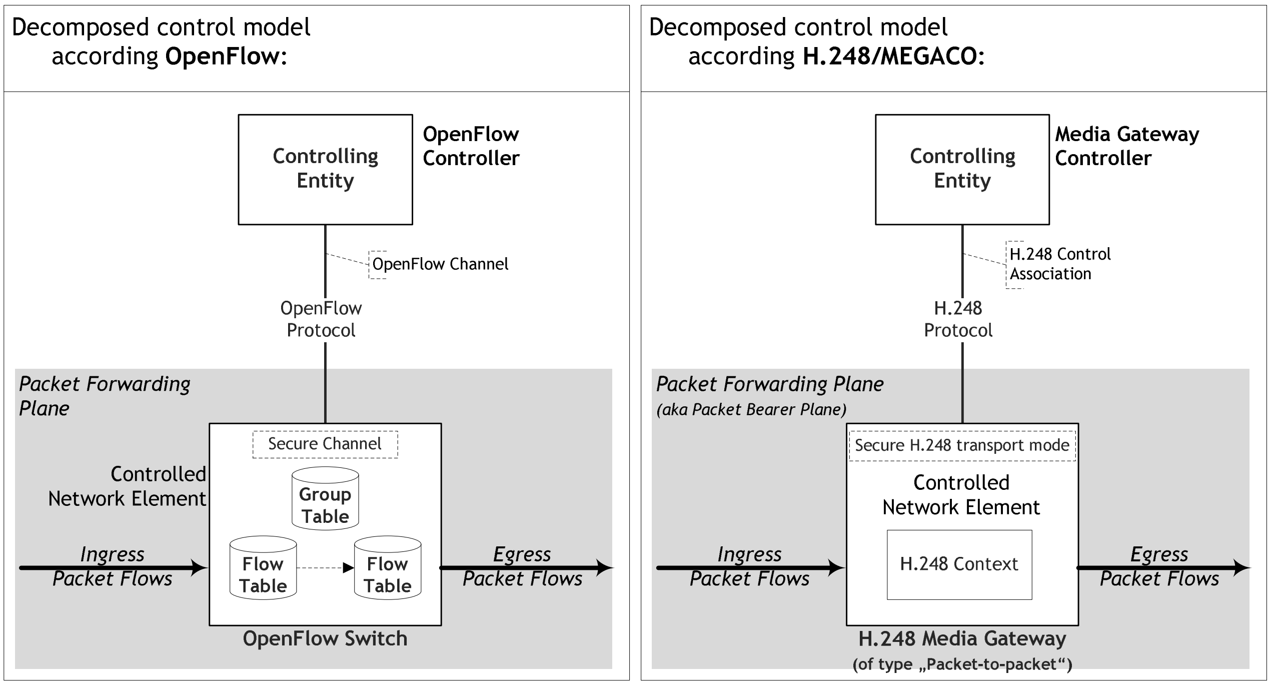

Figure A.1: Decomposed control models: OpenFlow vs. H.248

Any further evaluation implies firstly the availability of a specification of detailed requirements for the "vertical SDN control" interface, typically "SDN use case dependent", which is out of scope of the present document.

The Technical Report Draft TR H.Sup.OpenFlow [i.21] will provide more information concerning "OpenFlow vs H.248" (Note: still work in progress in ITU-T).

A.1.5 SDN Architecture Overview

In SDN, the data forwarding plane remains in the network elements while the control plane is typically provided by an SDN controller.

Each SDN controller can control multiple SDN network elements. SDN controllers are software applications that may be running on virtualised architecture. SDN controllers may be centralized, with a single controller for the network, or in a hierarchy, with client/server relationships between SDN controllers. There are some cases where distributing the SDN controller functions may provide better scalability; in that case, virtualisation of the controller functions may be useful as the SDN controller-based VNFs may be deployed dynamically, on demand, and can migrate following the associated workload.

When a centralized SDN controller topology is required, the benefits of virtualisation of an SDN controller include the ability to better scale up and scale out. The former could be used to add more resources to an existing SDN controller, for example additional storage resources. The latter could be used when an SDN controller is implemented as a set of collaborating VNFCs and additional VNFC instances are added to respond to the increased workload.

From the deployment and implementation prospective, it may be beneficial to implement an SDN controller as a VNF. SDN controllers can be implemented as part of service offering, such as NFPaaS.

Similarly, SDN network elements can be implemented as software running on general purpose platforms in a virtualised environment; as such they can be VNFs as well. SDN network elements can be hybrid and support both SDN and non SDN models. SDN may be enabled on an SDN network element VNF by uploading an SDN driver when deployed or on demand after deployment.

SDN controllers make forwarding decisions and upload packet forwarding tables on the SDN network elements via an open interface, for example OpenFlow. SDN controllers also discover the underlying network topology and ensure consistency of packet forwarding tables. SDN controllers offer northbound APIs to the applications or client SDN controllers. For example, an SDN controller sets up end-to-end paths, typically with overlays, at the edge of the network and then provides northbound APIs that enable 3rd parties to develop their own path instantiation applications.

Network applications can be included the SDN controller, provided by the SDN controller vendor, or may be provided by 3rd party if the SDN controller offers an SDK. Network applications can include SDN controller functions or they can interact with the northbound API of an SDN controller. Most common northbound APIs are RESTful APIs. Network applications, including service chaining applications, can themselves be VNFs. When an SDN controller is implemented as a VNF, the network element which it controls can be VNFs or PNFs. The SDN controller and those network elements should be within one VNFFG.

FCAPS management systems for the SDN environment, including SDN network elements and SDN controllers, are typically provided by the SDN vendors as EMSs. EMSs can be provided as PNFs or software running on a virtualised environment, i.e. VNF.

When an SDN controller or an SDN network element is implemented as a VNF, its management is not different than other VNFs: it should fit within the NFV architecture, integrate with NFV management and orchestration and run on the NFV Infrastructure.

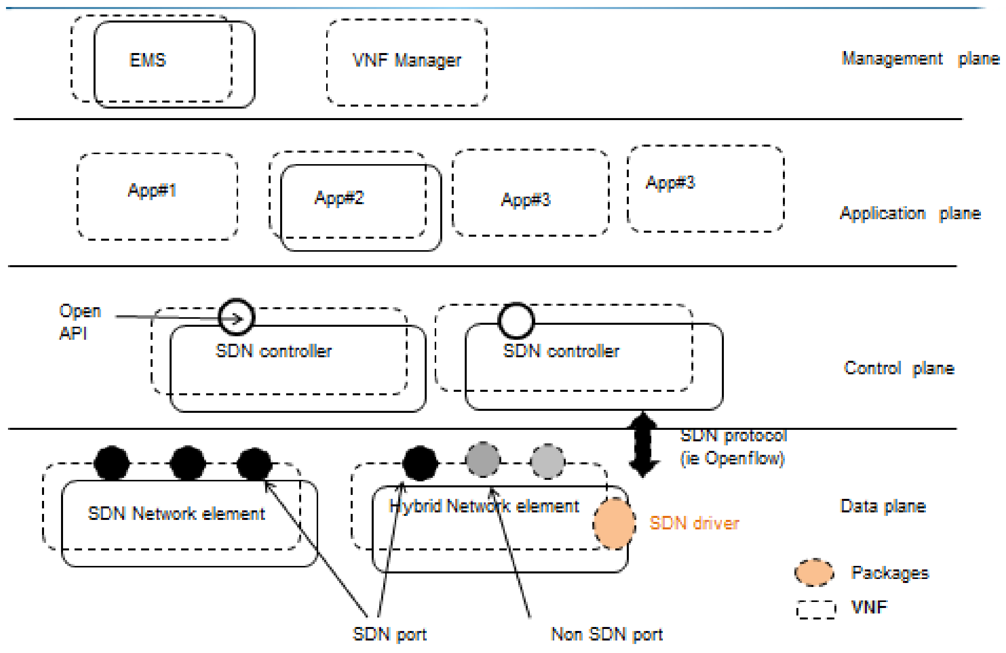

Figure A.2: SDN Architecture and VNF mapping

A.2 SDN in NFV Architecture

A.2.1 Overview

SDN environment may be virtualised or non-virtualised.

The NFV architecture supports both models and would need to interact with these different SDN components in a seamless manner.

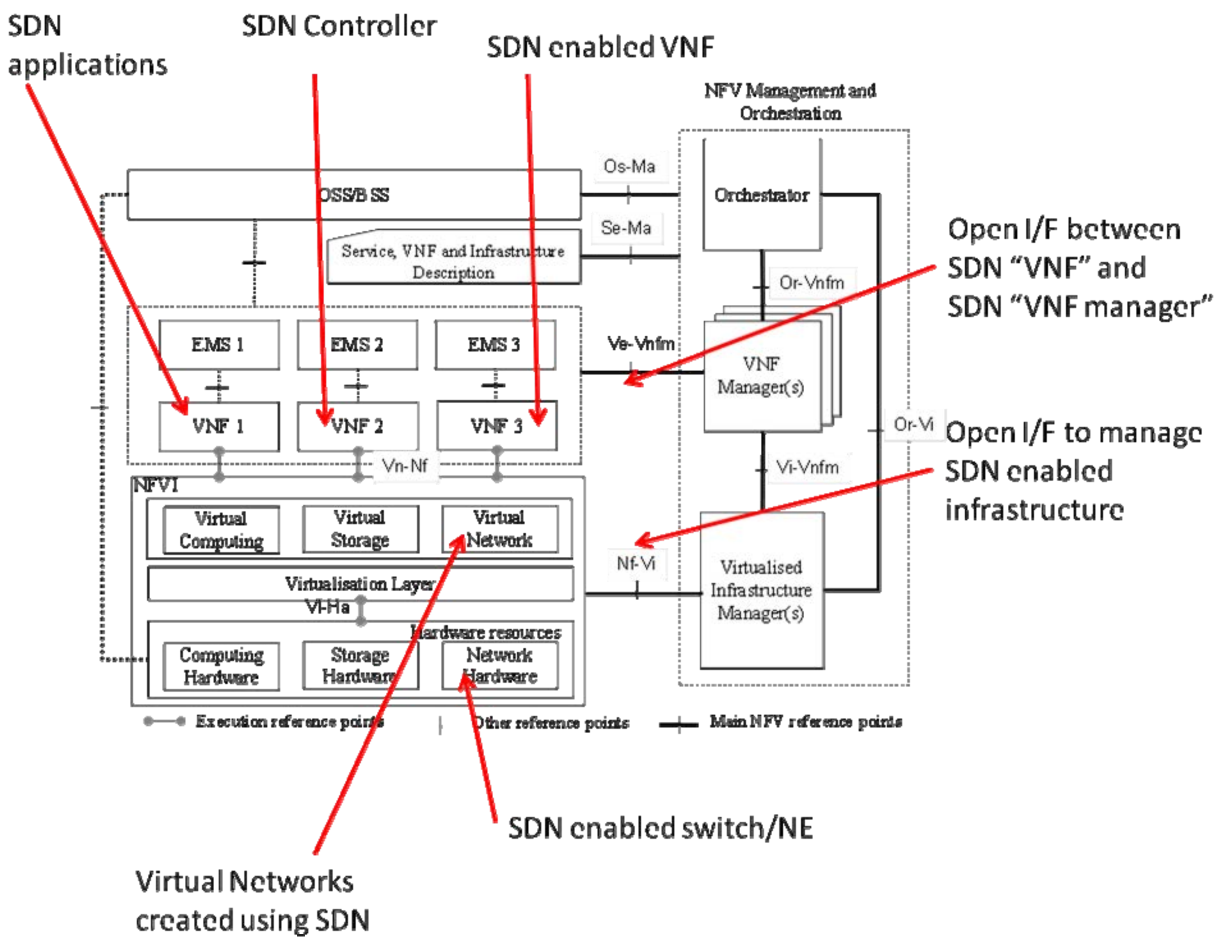

Figure A.3: Mapping of SDN components with ETSI NFV Architecture

Description of the arrows:

- SDN enabled switch/NEs include physical switches, hypervisor virtual switches and embedded switches on the NICs.
- Virtual networks created using an infrastructure network SDN controller provide connectivity services between VNFCIs.
- SDN controller can be virtualised, running as a VNF with its EM & VNF Manager. Note that there may be SDN controllers for the physical infrastructure, the virtual infrastructure and the virtual and physical network functions. As such, some of these SDN controllers may reside in the NFVI or MANO functional blocks. Note that this is not shown in this figure as this covered in the respective GS (ref MANO and INF Network GS).
- SDN enabled VNF includes any VNF that may be under the control of an SDN controller, e.g. virtual router, virtual firewall, etc.
- SDN applications, for example service chaining applications, can be VNF themselves.
- Nf-Vi interface allows management of the SDN enabled infrastructure.
- Ve-Vnfm interface is used between the SDN VNF (SDN controller VNF, SDN network functions VNF, SDN applications VNF) and their respective VNF Manager for lifecycle management.
- Vn-Nf is allowing SDN VNFs to access connectivity services between VNFCIs.

Typical NFV requirements apply to SDN environments:

- When SDN controller is virtualised, and is a VNF, the "SDN controller" VNF EM should support the ETSI NFV Ve-Vnfm interface - It should not use this interface to communicate with SDN network elements. The connectivity to the SDN network elements is achieved through the Vn-Nf interface providing connectivity services.
- SDN component VNF vendors should provide VNF Descriptors and all the metadata required for the NFV orchestrator.

A.3 ETSI NFV Use Case and SDN

A.3.1 L2 Transparent Network Service Chaining with Traffic Steering

A.3.1.1 Problem Description

A.3.1.1.1 L2 Transparent Network Services

L2 transparent network services such as L2 security services are normally used in networks to avoid any changes to the IP addressing related network configuration. In physical world, L2 transparent network services appear as L2 bridges to rest of the network and thus avoids the need for IP address changes in rest of the network elements. L2 transparent network services do their operations while bridging the packets. Being bridges, they do not need to run any unicast or multicast routing protocols, thereby making them simpler to configure and manage.

A.3.1.1.1.1 Physical Server Deployments

Introducing L2 transparent network devices in physical world is simple to understand.

Let us assume that there are set of physical servers connected to a L2 switch as shown in figure A.4.

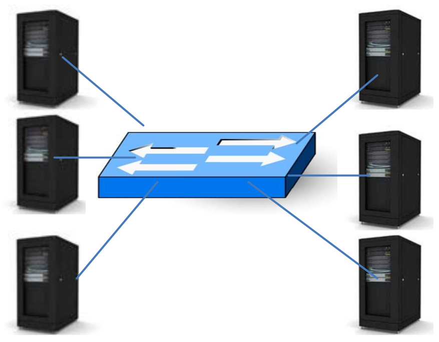

Figure A.4: Example Physical Server Deployment

After a while, network administrator decides to add security between left and right side of servers. L2 security services would be a right choice as it does not require any IP addressing related configuration changes in the rest of the network. Introduction of security services could be drop-in as shown below. It requires addition of one more switch (assuming VLAN separation and associated L3 reconfiguration is to be avoided) and introducing service chain with two security devices.

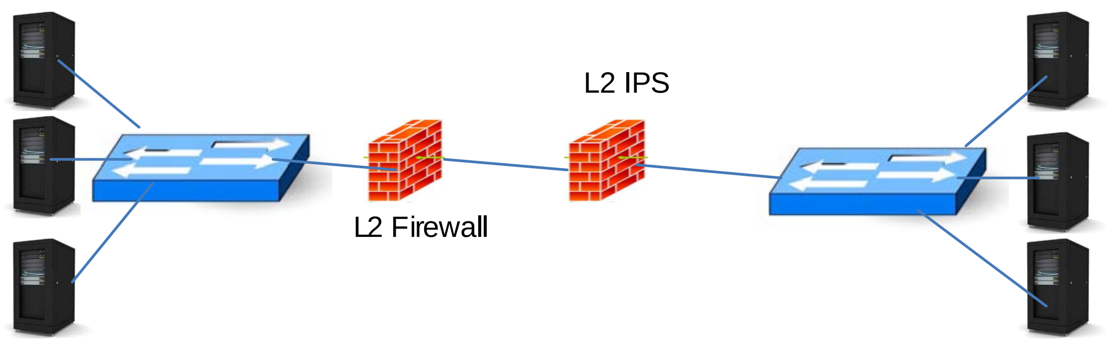

Figure A.5: Example L2 Firewall and L2 IPS

In figure A.5, L2 transparent firewall and IPS are added between two sets of servers.

There is another choice network administrator has to introduce network services. Some L2 switch solutions provide inbuilt network services. One could replace the switch with more intelligent and integrated switch as shown below. Typically, vendors provide this capability by providing network service blades in their switch chassis.

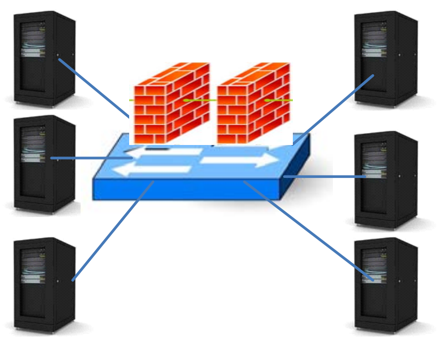

Figure A.6: Example L2 Switch Solution

Performance scaling is one of the important aspects in network deployments. In physical world, this is not solved very well. There are two approaches followed normally. One method to replace the existing network service devices with more powerful devices and second method is to scale-out with external special purpose load balancers as shown below. In the picture below, it is assumed that performance scaling is required on IPS service. Unlike Server clustering where the clustering can be realized using one load balancer, network service clustering requires two load balancers, one on each side of the cluster. Also, these two load balancers need to ensure that the packets belonging to a TCP/UDP connection go through the same network service device. Many network security services are stateful in nature. They can do better security analysis of the traffic only if all the packets belonging to a connection are seen. Hence, these two load balancers have to have a common understanding to achieve this and hence they need to special kind of load balancers.

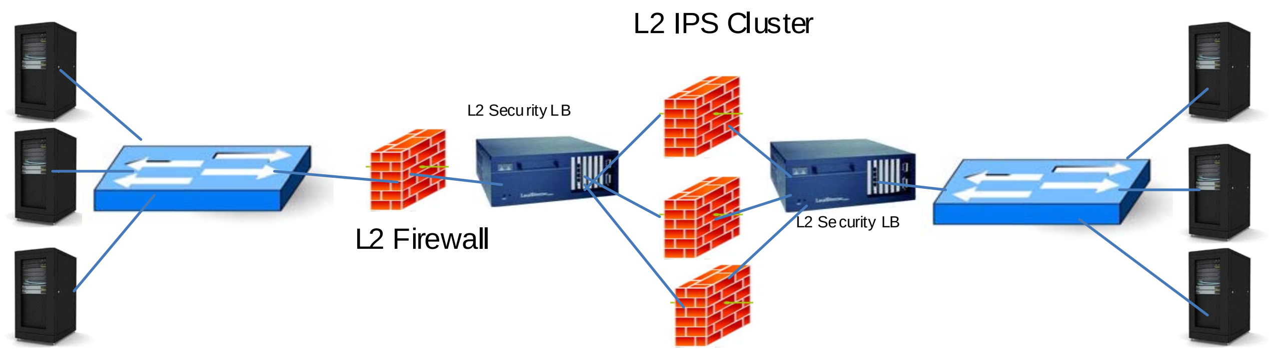

Figure A.7: Example L2 Simplify Transparent network Service

Though L2 transparent network services simplify some network configuration, there are few challenges as shown above. Challenges with physical deployments are:

- Fixed Service Chain: All types of traffic would go through the network devices that need to be in the chain. Hence, all devices in the path or cluster of devices in the path will have to be able to take up the traffic load.
- Service Insertion or Deletion: Addition of new network service (or deletion of existing network service) could take days as it requires rewiring the cables (or reconfiguring L2 switches to add new VLANs). Since service insertion and deletion affects the existing traffic, this kind of change typically done off-peak hours. With 99,9999 % kind of reliability, no disruption may be acceptable.
- Complicated scale-out architecture: Scale-out requires two sets of load balancers on either side of the device clusters. It increases the cost of deployment:
  - Need for additional load balancers, possibly, for each network service cluster in the chain.
  - Additional load balancers for high availability.
  - Powerful load balancers as all the traffic passes through them.

Also increases the latency of the packets as the packets need to traverse through additional devices.

A.3.1.1.1.2 Virtual Server Deployments

In virtual deployments, servers are run as virtual machines on compute nodes. Few important points to note are:

- Multiple servers could be instantiated on a physical server (compute node).
- Servers of various privileges or levels can be instantiated on the same compute node. For example, Web servers and database servers could be instantiated on a compute node.
- Servers can be live migrated from one compute node to another.

How would security be applied in virtual environment?

If physical security devices are to be used, then it becomes necessary that virtual machines of various types are instantiated in separate physical serves to enable placement of security devices (assuming VLAN separation and associated L3 reconfiguration is to be avoided). For example, if L2 security is required between web server VMs and DB VMs, then web servers and DB may have to be instantiated on separate physical server and apply security as shown below.

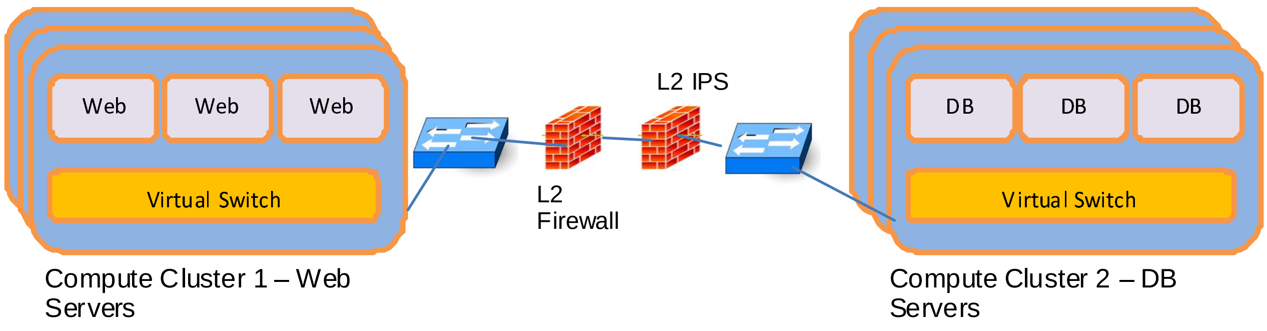
Web
Web
Web
Virtual Switch
L2 IPS
L2 Firewall
DB
DB
DB
Virtual Switch
Compute Cluster 1 – Web Servers
Compute Cluster 2 – DB Servers

Figure A.8: Example Virtual Server Deployments

This type of restriction poses various challenges:

- Dedication of set of compute nodes for various types of servers could lead to improper balancing of compute nodes. It also requires constant monitoring of multiple compute clusters to determine whether to add new compute nodes and remove some compute nodes from various clusters.
- Network Service devices supporting overlay technologies: Overlay technologies such as VxLAN and NVGRE are increasing being to realize virtual networks instead of VLANs due to multiple limitations of VLAN technology. Any network service that is introduced between the clusters shall also support overlay networks and this leads to configuration of overlay network not only in orchestration layers, but also in these network service devices. One should ensure that configuration consistency is maintained. Since there are various dashboards, one for each network devices, this could be complex and can lead to configuration errors.
- Overlay with IPSec: VxLAN over IPSec kind of overlay networks encrypts the packets when they leave the compute node and the receiving compute node decrypts the packets at NFVI layer. Network service devices in the middle require packets in clear. It means that these network devices shall also support IPSec functionality. It results into more configuration by the administrators and could lead to more configuration mistakes.

It also inherits similar challenges as in physical deployments related to network service insertion/deletion and scaling-out.

A.3.1.2 Solution Description & Relationship to SDN/OF

Introducing network services in virtual deployment should be simple, fast and less error prone. There should be no restrictions on server VM placement and server VM movement and no dependency of virtual network technologies.

This use case utilizes two technologies realize this solution. One is NFV technology where network services are also deployed as VMs and another is SDN/OF technology which can be used to steer the inter server-VM traffic across multiple network service VMs.

A.3.1.2.1 SDN/OF virtual switches

Modern hypervisors introduced virtual switches that are controllable from outside. Linux adopted SDN/OF protocol based virtual switch as part of its virtual environment. Each compute node has one virtual switch. All the traffic from/to virtual machines on the compute node passes through these virtual switches. Unlike older and proprietary virtual switches, SDN/OF based virtual switches are programmable from external controller. Since all the traffic across VMs within a compute node or across compute nodes pass through programmable virtual switches, traffic steering application on top of a central controller becomes a possibility.

OVS (Open Virtual Switch) is a virtual switch implementation in recent Linux distributions. OVS supports OF 1.3 protocol, which allows external controllers to create a pipeline of packet processing using multiple tables. Each table can be programmed with multiple entries. These tables support any combination of match fields. OVS also supports many instructions/actions which enable multiple virtual network technologies including VxLAN, NVGRE and of course VLAN.

A.3.1.2.2 Role of SND/OF in Service Chaining Traffic Steering

OpenFlow based switches are enabling applications in central controller to implement traffic redirection capabilities including steering the traffic selectively across multiple network service VMs.

In physical deployments, introducing L2 network services is as simple as cutting the wire and inserting the network service devices. In virtual deployments, there is no wire to cut. Fortunately, SDN/OF based virtual switches help in inserting the network service VMs between server VMs. Following sub topics describe on how service chaining is achieved with and without scale-out.

Service Chaining

Figure A.9 describes the service chaining. The picture only shows the traffic flow, but the following configuration steps are expected to be done by administrator using orchestration tool such as Openstack. Configuration steps typically involve:

- Uploading network service VM images into orchestration tool. In this example, it is expected that 'firewall', 'DDOS prevention', "IPS', "DPI and "WAF" VM images are uploaded in the orchestration tool.
- Creation of network service chain.
- Adding network services to the chain. In this example, "firewall", "DDoS prevention", "IPS", "DPI" and "WAF" services are added to the chain.
- Creating bypass rules. In this example, web traffic is expected to be passed through Firewall, IPS and WAF services. And all other traffic is expected to be passed through Firewall, DDoS, IPS and DPI services. Hence, two bypass rules would need to be created:
    - Rule 1: Web traffic, bypass "DDoS", "DPI".
    - Rule 2: All traffic, bypass "WAF".
- Attaching the network service chain to a virtual network.

Once above steps are done, orchestrator is expected to instantiate network service VMs (one of each service) upon first application VM in that virtual network is brought up.

Orchestrator programs the SDN/OF controller with the above configuration.

Packet flow: For simplicity, following picture shows each VM on a separate compute node. In reality, multiple VMs can be instantiated on a compute node. Each compute node has OFLS (Openflow Logical switch) virtual switch. All virtual switches of compute nodes are configured to communicate with centralized SDN/OF controller. SDN/OF controller runs traffic steering application.

Step1: AppVM on left side initiates a HTTP connection to the AppVM on the right side. When the virtual switch on local compute node sees the 1st packet, it does not find any flow in its tables. Hence, it sends that packet to the SDN/OF controller via OF protocol as an exception packet.

Step 2: Traffic steering application in SDN/OF controller, refers to the service chain configuration associated with the virtual network. It also finds out the services to be applied on the traffic of this connection. Since the rule (Rule 1) indicates that DDOS and DPI services need to be bypassed for HTTP traffic, it chooses to create flows in the virtual switches where application VMs, "firewall", "IPS" and 'WAF" VMs are present. In this example, virtual switches A, E, B, C, D are populated with the flows. These flows are meant to redirect the traffic one to another. It, then, sends the packet back to the virtual switch.

Step 3: Once the flows are created, all packets of the connection between these two VMs are passed through the network service VMs.

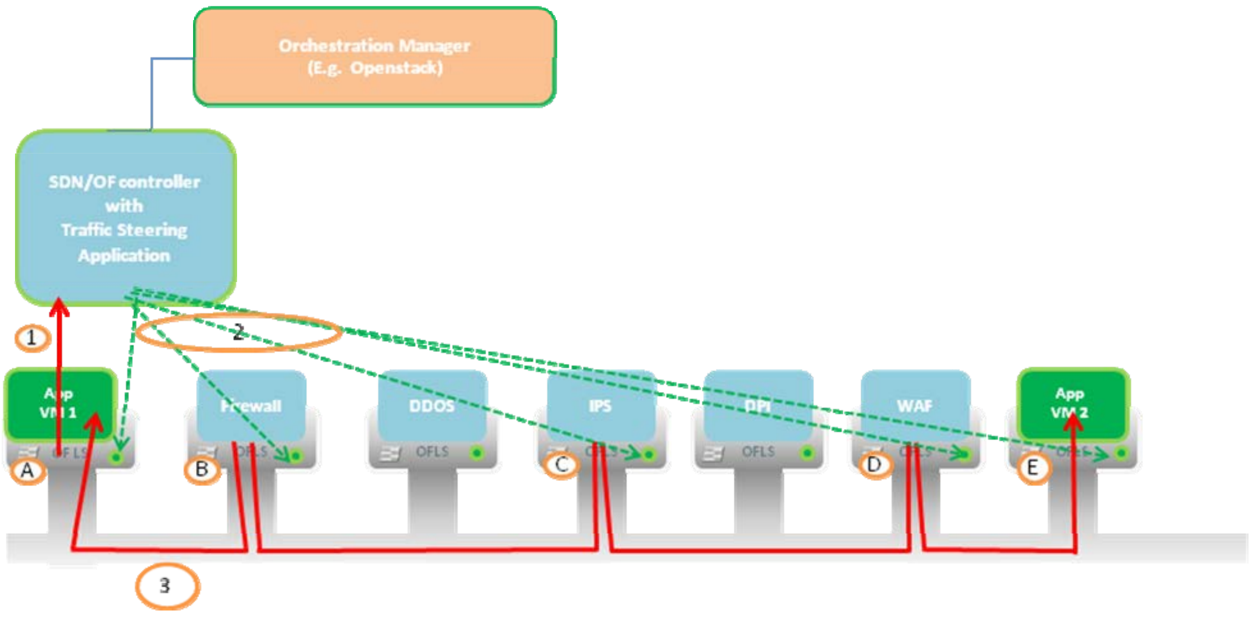

Figure A.9: Example SDN/OF Controller with traffic

Details of flows that are created as part of Step 2 by the SDN/OF controller:

Assumptions:

- AppVM1 IP address = I1 and AppVM2 IP address = I2.
- Compute node IP addresses A's address = O1, E's address = O2 B's address = O3, C's IP address = O4 and D's IP address = O5.
- HTTP Connection: TCP Source Port = 2000 and Destination Port = 80:
  - Client to Server 5 tuple are: I1, 2000->I2, 80 with TCP.
  - Server to Client 5 tuple is: I2,80->I1, 2000 with TCP.
- All VMs in each compute node are connected via port VM-p.
- Both the legs of network services are identified by VLAN100 and VLAN200.
- VxLAN based virtual network is used. VNI = 50.
- Flow based tunnelling mechanism is used.
- Network port in all virtual switches are denoted as N-p.

OF flows:

Traffic flows in virtual switch A:

- Flow 1 (Client to Server flow)
  - Match fields:
    - Input Port = VM-p.
    - Source IP = I1, DestinationIP = I2, Protocol = TCP, Source port = 2000 and Destination Port = 80.
  - Instructions/Actions: Next hop if virtual switch B and firewall leg VLAN 100:
    - Set fields: Tunnel_ID = 50 (VNI), Tunnel Destination IP = O3.
    - Push VLAN - VLAN100.

        - Output: N-p.

- Flow 2: (Server to Client flow):
    - Match fields:
        - Input port = N-p.
        - Source IP = I2, Destination IP = I1, Protocol = TCP, Source Port = 80 and Destination port = 2000.
    - Instructions/actions:
        - Output: VM-p.

Traffic flows in virtual switch B: Here four flows are required - One for receiving client to server traffic from the network to give it to the firewall, second flow is for receiving the traffic from the firewall VM to send it onto the network towards next service, third flow is for receiving server-to-client traffic from the network to give it to the firewall and fourth flow is for receiving the packet from firewall to send it onto next service/appvm.

- Flow 1 (Client to Server traffic from network):
    - Match fields:
        - Input Port = N-p.
        - VLAN = VLAN100.
        - Source IP = I1, DestinationIP = I2, Protocol = TCP, Source port = 2000 and Destination Port = 80.
    - Instructions/Actions:
        - Output = VM-p.
- Flow 2 (C-S traffic from firewall VM):
    - Match fields:
        - Input port = VM-p.
        - VLAN = VLAN200.
        - Source IP = I1, DestinationIP = I2, Protocol = TCP, Source port = 2000 and Destination Port = 80.
    - Instructions/Actions: Next hop if virtual switch C and IPS leg VLAN 100:
        - Set fields: Tunnel_ID = 50 (VNI), Tunnel Destination IP = O4.
        - PopVLAN - VLAN200.
        - Push VLAN - VLAN100.
        - Output: N-p.
- Flow 3: (S-C traffic from network):
    - Match fields:
        - Input Port = N-p.
        - VLAN = VLAN200.
        - Source IP = I2, Destination IP = I1, Protocol = TCP, Source Port = 80 and Destination port = 2000.
    - Instructions/actions:
        - Output = VM-p.

- Flow 4 (S-C traffic from firewall VM):
  - Match fields:
    - Input port = VM-p.
    - VLAN = VLAN100.
    - Source IP = I2, Destination IP = I1, Protocol = TCP, Source Port = 80 and Destination port = 2000.
  - Instructions/Actions:
    - Set fields: Tunnel_ID = 50 (VNI), Tunnel Destination IP = O1.
    - PopVLAN - VLAN100.
    - Output: N-p.

Traffic flows on virtual switch C and D look similar to the flows on virtual switch B.

Traffic flows in virtual switch E:

- Flow 1 (Client to Server flow):
  - Match fields:
    - Input Port = N-p.
    - Source IP = I1, DestinationIP = I2, Protocol = TCP, Source port = 2000 and Destination Port = 80.
  - Instructions/Actions: Next hop if virtual switch B and firewall leg VLAN 100:
    - Output: VM-p.
- Flow 2: (Server to Client flow):
  - Match fields:
    - Input port = VM-p.
    - Source IP = I2, Destination IP = I1, Protocol = TCP, Source Port = 80 and Destination port = 2000.
  - Instructions/actions:
    - Set fields: Tunnel_ID = 50 (VNI), Tunnel Destination IP = O5.
    - Push VLAN - VLAN200.
    - Output: N-p.

Scale-out

SDN/OF controller can also acts as implicit load balancer, thereby avoiding any external load balancers to increase the scale of network services. In the following picture, there are three IPS service VMs instantiated and 2 WAF VMs are instantiated. SDN/OF controller, for every new connection, it chooses the IPS VM and WAF VM that is least loaded. Accordingly, it programs the flows in appropriate OF logical switches.

In figure A.10, red connection traffic is passed through one IPS VM and blue connection traffic is passed through another IPS VM.

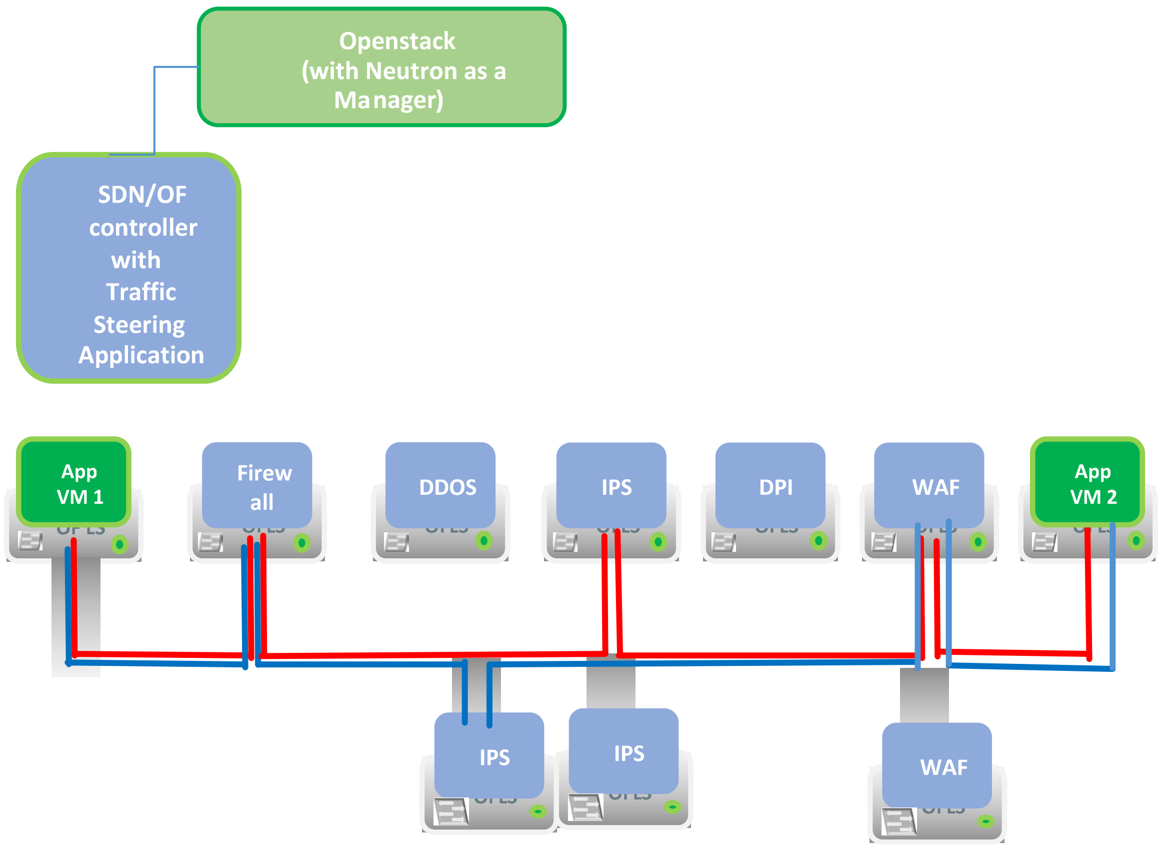

Figure A.10: Example Traffic flows

A.3.1.3 Requirements to Management and Orchestration

NFV Management & Orchestration requires SDN/OF controller within VIM. SDN/OF controller needs to talk to the virtual switches of compute nodes to program flows. Hence there is a need for an interface point between SDN/OF controller (VIM) & compute nodes (NFVI). Or-Vi and Nf-Vi would serve this purpose.SDN/OF controller also needs to be informed of network service chain configuration. Hence there is a need for an interface point between orchestration system (VNFM) and SDN/OF controller system (VIM). Vi-Vnfm and Ve-Vnfm would serve this purpose.

NFV management and orchestration needs to be able to satisfy the following requirements:

- Service Chain Management:
  - Ability to manage (Create/Delete/Update) network service chains.
  - Ability to manage (Add/Delete/Update) network functions in the chain - Service Insertion/Deletion [MANO service graph update workflows].
  - Mitigate service interruption to existing traffic when VNF instance fails in the chain.
  - Ability to define traffic rules to bypass network functions in the chain.
- Management of network service chain with virtual networks:
  - Ability to associate multiple network service chains to a network.
  - Ability to define traffic rules for selecting the chain out of multiple chains.
- Scale-Out:
  - Ability to bring up/down multiple network function VMs on demand basis.
  - Ability to balance the traffic across multiple VMs of a network function with no special load balancers.

- Fast Path:
  - Ability for network service VMs to offload connections to some forwarding entity.

One orchestrations system to deploy both server and network service VMs.

A.3.1.4 Gap Analysis with ONF

OF 1.3 support multiple tables. Traffic steering application can use few tables to install traffic redirection flow entries. A few extensions in OF 1.3 are required to support flow based tunnelling mechanism. Though those extensions are not part of the OF 1.3 specifications, they are widely implemented in OVS. It is required that these extensions are standardized.

Annex B (informative): De/composition Study

There are a plethora of decomposed network elements related to the network function category "gateways", in "all packet" (or "all IP") network infrastructures. A well-known decomposition model follows the signalling-media plane separation according to Recommendation ITU-T H.248.1 [i.20]. Clause B.1 provides such an example with scope on network function "media server". The underlying 3GPP protocol is the ETSI TS 129 333 [i.34] H.248 Mp profile.

B.1 MRF IMS Use Case

This clause is aimed to describe an IMS (IP Multimedia Subsystem - 3GPP) MRF use-case instantiated as a virtualised Network Function VNF. This example is particularly relevant as it is part of IMS, listed as one of NF top priority for operators to be virtualised, and it illustrates a case where typically MRF-P and MRF-C have different resource needs.

MRF is defined by 3GPP as a Media Resource function (ETSI TS 123 228 [i.11]). It is composed of 2 standardized elements MRF-C controller, and MRF-P processor.

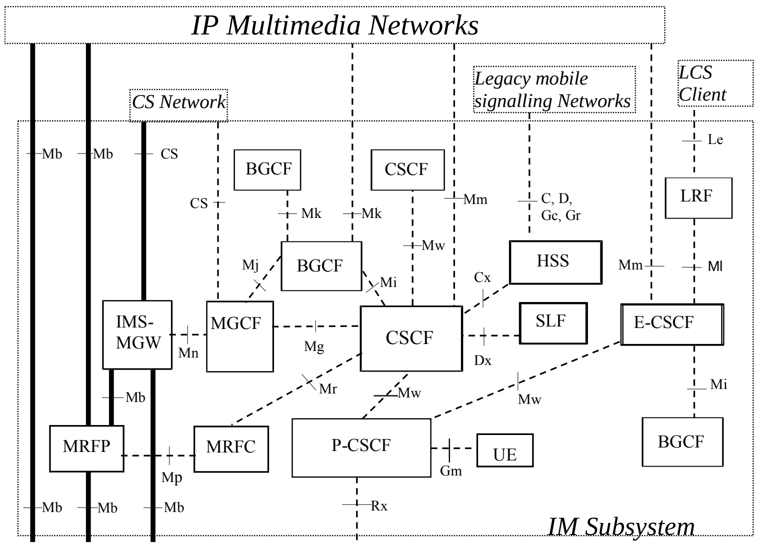

Figure B.1: Reference Architecture of the IP Multimedia Core Network Subsystem, [i.1]

In Rel 8 was also introduced a MRB, Media Resource Broker. Further details about the MRB and relationships between the MRF and other IMS components can be found in ETSI TS 123 218 [i.12].

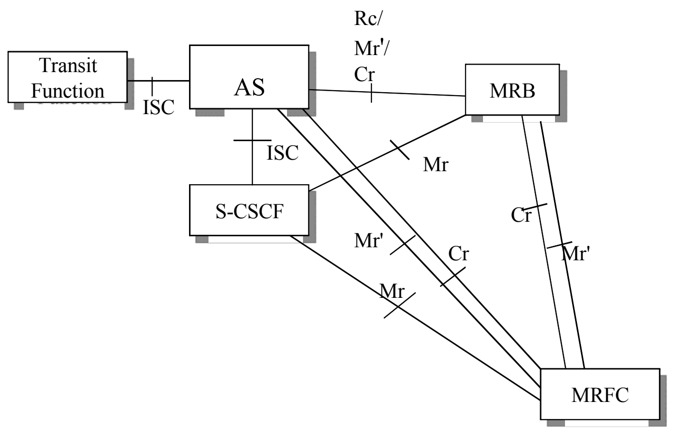

Figure B.2: Extract from 3GPP 23.218 describing the MRB with MRF (MRF-C)

In this use case, IMS MRF (with MRF-C and MRF-P) and an MRB are considered.

B.1.1 Functional Description

Figure B.3 depicts an IMS MRF instantiation.

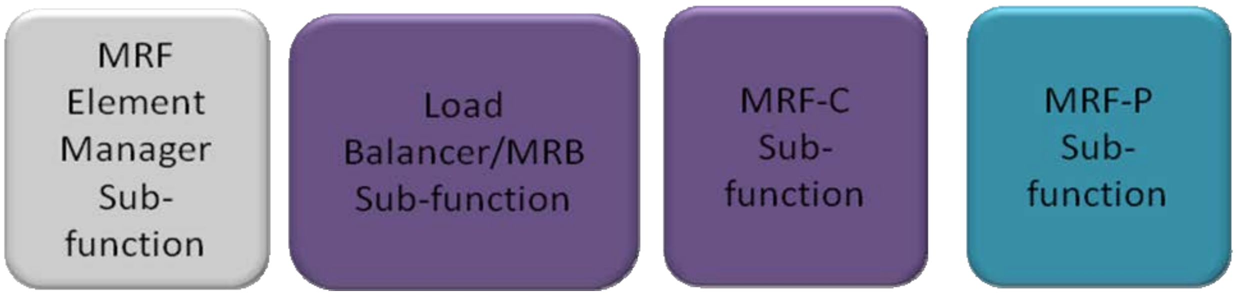
MRF Element Manager Sub-function
Load Balancer/MRB Sub-function
MRF-C Sub-function
MRF-P Sub-function

Figure B.3: IMS MRF instantiation in a virtualised environment

Figure B.3 depicts the 4 main sub-functions for this IMS MRF instantiation.

The MRF-C represents the controlling function of the MRF; it is a signalling function that terminates the Mr, Mr' and Cr Control interfaces and drives the MRF-P functional entity. The MRF-C can also provide features like Audio/Video prompts management, live monitoring and off-line reporting of MRF operations.

The **MRF-P** represents the multimedia processing function of the MRF; it connects to the Mp data interface that is used to carry user plane traffic (e.g. audio and video samples over RTP and its control part RTCP). It is a data plane function, although it may have to process signalling-like protocols as well (e.g. floor control commands using BFCP, messaging over MSRP).

The **Load Balancer or Media Resource Broker (MRB)** is aimed to load balance the incoming SIP Session towards several MRF-C instances. It is a signalling function that supports the Mr, Mr', Cr and Rc IMS interfaces.

The **MRF Element Manager** is in charge of:

- Configure the different sub-functions.
- Manage the interactions between these sub-functions.
- Request and Collect metrics from outside system and compute, i.e. number of Call Attempts per Second per MRF-C instance, time to established SIP Session, time to fetch Resources like Audio/Video prompts, etc.

- Implement specific logic, like Elasticity (timely match of demand and supply) for instance and request resources to be allocated on the virtualised infrastructure, i.e. Add/Remove resource to a Virtual Machine like vCPU or vRAM, or to Add/Remove a Virtual Machine instance of a MRF-C/MRF-P instance, etc.

Note that mapping the MRF-C and MRF-P on two different VNFCs or grouping them in one VNFC is a design choice. The MRF is particularly interesting because it combines signalling and user/data plane functions. Typically when handling voice and video communications, MRF-P is processing multimedia samples periodically in less than 10 ms. Moving to a virtualised environment the Digital Signal Processing is moving from specialized HW to software DSP paradigm, it requires specific processing capacity while defining the underlying virtualised infrastructure or evolving that capacity dynamically.

In order to execute these functions on multi VNF Provider environment, HW and hypervisor agnostic, an open standard APIs on top of the INF is recommended, to avoid having to implement multiple proprietary I/F to request proper computing capabilities, and collect real time metrics to ensure service scalability and continuity.

B.1.2 Location of FBs within the NFV Architecture

Figure B.4 represents a virtualised MRF comprising the sub-functions identified in clause B.1.1 and shows that, as any other VNF, the virtualised MRF interacts with other elements of the NFV architecture, such as NFV Management and Orchestration, NFV Infrastructure, OSS/BSS and legacy systems.

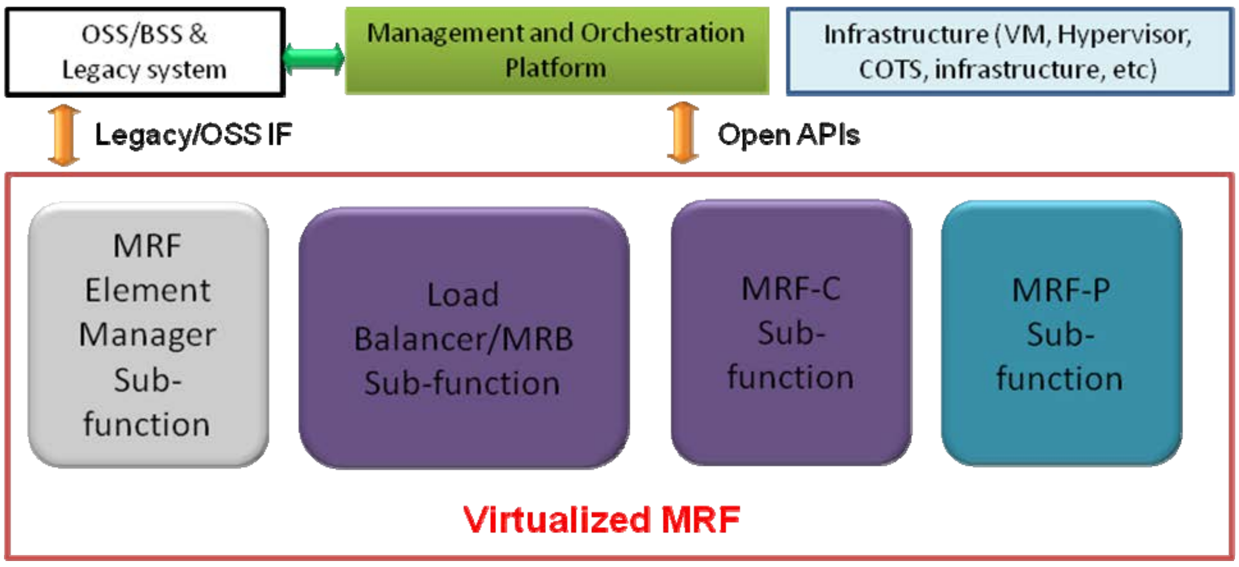
OSS/BSS & Legacy system
Management and Orchestration Platform
Infrastructure (VM, Hypervisor, COTS, infrastructure, etc)
Legacy/OSS IF
Open APIs
MRF Element Manager Sub-function
Load Balancer/MRB Sub-function
MRF-C Sub-function
MRF-P Sub-function
Virtualized MRF

Figure B.4: MRF interactions with OSS/BB, NFV Management & Orchestration and NFV Infrastructure

B.1.3 New Interfaces

A virtualised MRF instantiation requires interfaces with both Management & Orchestration and with the virtualised infrastructure.

Requirements to Management & Orchestration

A virtualised MRF instantiation will require interaction with Management and Orchestration along its life cycle for:

- The design of a virtualised MRF for a given deployment description of the VNFCs and requirements to deploy, run, delete/phase out.

- The deployment of a virtualised MRF - there may be different use cases, for instance deployment of a complete new MRF instance and all the other VNF Instances it depends on or deployment leveraging already deployed VNF Instances, i.e. leveraging an existing load balancer.

- The run time of a virtualised MRF:
    - A live MRF requires some metrics to ensure quality of service and meet SLA. These metrics are predefined in advance with the management and orchestration environment and either pushed with a given frequency by management & orchestration to the network function (or to the EM of the network function) - or they can be requested on demand by the network function to management & orchestration, i.e. provide past 2 days critical alarms.
    - A live MRF also requires some scalability management from Management & orchestration: scale up, scale down, scale in and scale out, etc.
    - Upgrades could be required: deploy a new image, new version.
    - Upgrades in terms of functionality relying on infrastructure, i.e. move to HA (more HW, new SW).
    - Etc.
- The deletion of an MRF instance: it will also require passing that request to the management and orchestration to remove either a given VNFC instance (if the MRF sub-function is mapped to a VNFC) without service interruption, or an entire VNF instance (if the MRF sub-function is mapped to a VNFC), with or without the Element Management.

Requirements to Infrastructure

The requirements on the infrastructure are the following:

- Allocate the resources to the VNF's virtual machines:
    - Allocation of the guaranteed and non-guaranteed virtual CPU and virtual RAM.
    - Allocation of IP network resources like IP subnets, VLANs and IP addresses range that fits the bandwidth and QoS requirements (Packet loss, Jitter, latency). In addition to required IP services like multicast/IGMP enablement, DNS services, firewall holes opening, etc.
    - Virtual storage capacity fitting the expected IO Read/Write bandwidth.
- Enable the definition of group of VMs together with their affinity and anti-affinity rules.
- Change the allocation policy for scale-in and scale-out scenario.
- For each allocated virtual machine, a set of resource Key Performance Indicators' probes need to be activated with corresponding thresholds. The requirement is to inform the VNF (or VNF EM) in case of resource shortage.
- Provide the capability to execute operations on the VM like migrating live a VM from one HW resource to another HW resource, to gracefully stop a VM, and delete the VM instance.

Requirements on OSS/BSS & legacy systems

Virtualised MRF instantiation requires interface with existing OSS/BSS & legacy systems.

B.1.4 VNF Identification

This clause is an illustrative use case of the mapping of the sub-functions above to VNFs and VNFCs. For each identified VNF, a VNF descriptor (VNFD) will be defined with information described in functional terms split in deployment and operational information, addressing the lifecycle of the VNF.

It is not the objective of this clause to provide a recommendation for the mapping of 3GPP functions to VNFs and VNFCs. This is just an example of a possible mapping for illustrative purpose.

Note that depending on VNF Provider choice, the mapping might be slightly different. The VNF mapping proposed here can be considered as a typical one. It is not the goal of the present document to cover all possible VNF mapping use cases.

Regarding life cycle of the VNF, it is expected that existing standards or specification are leveraged, such as Cloud usage model defined in ODCA [i.10].

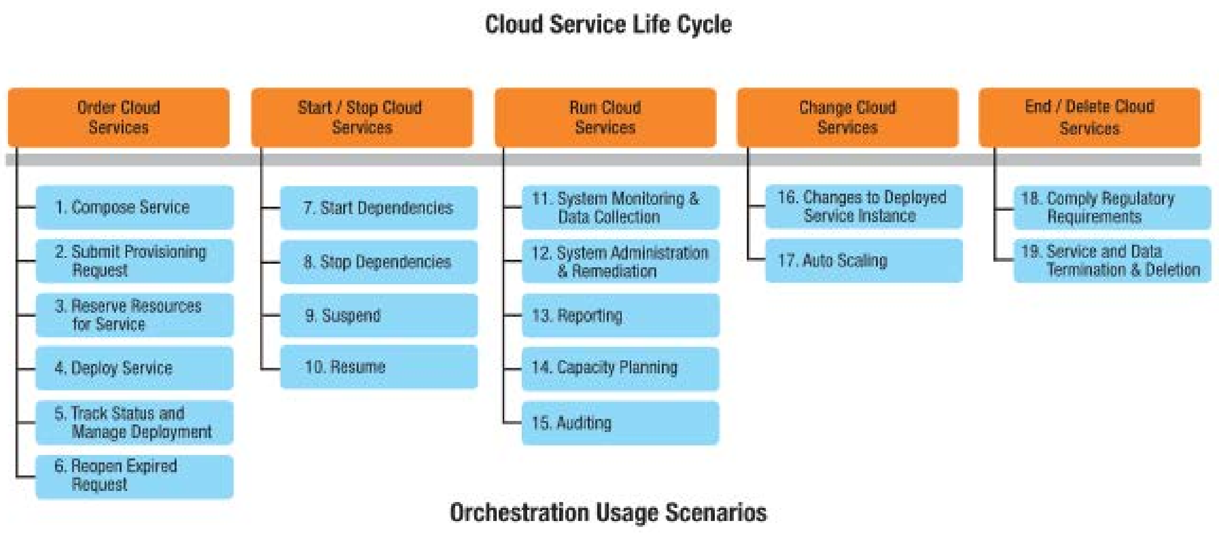
Cloud Service Life Cycle
Order Cloud Services
Start / Stop Cloud Services
Run Cloud Services
Change Cloud Services
End / Delete Cloud Services
1. Compose Service
2. Submit Provisioning Request
3. Reserve Resources for Service
4. Deploy Service
5. Track Status and Manage Deployment
6. Reopen Expired Request
7. Start Dependencies
8. Stop Dependencies
9. Suspend
10. Resume
11. System Monitoring & Data Collection
12. System Administration & Remediation
13. Reporting
14. Capacity Planning
15. Auditing
16. Changes to Deployed Service Instance
17. Auto Scaling
18. Comply Regulatory Requirements
19. Service and Data Termination & Deletion
Orchestration Usage Scenarios

Figure B.5: Cloud Service Life Cycle [i.10]

B.1.4.1 MRB

The MRB can be considered as a separate VNF as it can be deployed on its own in some case and it is also possible that MRB and MRF might come from different VNF Providers for the same project.

This is illustrated in figure B.6.

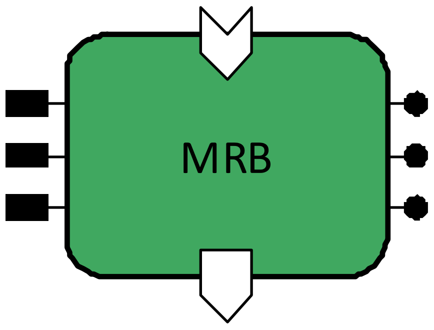

Figure B.6: MRB VNF

Deployment requirements

- The MRB uses an Active/Standby redundancy model. In case of Active MRB VNF failure, it is detected by the Standby MRB VNF that becomes active (with IP address takeover). There is no impact on established calls (with UDP transport).
- VM number: 2 instances. A minimum of 2 is required for availability and it is not needed to deploy more than 2 instances.
- VNF descriptor would also indicate also the number of vCPU needed and vRAM needed.
- For redundancy, anti-affinity rule to force to deploy the 2 instances on 2 VMs located on different physical hardware resources to augment the overall availability of the solution.
- Good network connection (highly Available IP network to guarantee 4 nines and more: no more than one Ping lost in case of network failure, packet Loss less than 0,5 %, round trip delay less than 120 ms, jitter delay less than 40 ms) between MRB instances.

Operational requirements:

- Support of typical processes in support of the VNF state transitions instantiated,, configure, start, stop and terminate.

Relationships

MRB has 2 external relationships:

- Relation MRB - MRF-C.
- Relations MRB - S-CSCF. An S-CSCF should be present in the overall solution, but it is not part of the MRB VNF and it might not even be a virtualised component. It would still be useful for NFVO to have an indication of this external dependency in the VNF descriptor.

For those 2 relationships, the requirements are a good network connection with the same parameters as described above.

B.1.4.2 MRF

As the MRF-C and MRF-P split in two sub-functions, the vendor can select to map to the virtual deployment can lead the MRF-C and MRF-P to be logically grouped together on the same VM. For this use case, MRF-C and MRC-P are considered as deployed together. Note that other mappings are possible like separate VNFCs for MRF-C and MRF-P, but the choice made in this use case is a typical one.

With the assumption that 1 VNFC should correspond to 1 VM, then MRF-C and MRF-P are combined for this use case in a single VNFC called MRF-C+P.

MRF-C also needs some external storage (RDBMS or SAN) that will have its own redundancy model. So while at the previous level, there is only a single MRF-C sub-function, at this level, MRF Storage is modelled also as a separate component. Note that MRF Storage might be further decomposed, for instance by introducing a load balancer as a front end to it, but for sake of simplicity, MRF Storage is kept as a single VNFC.

The MRF can be considered as a separate VNF for the same reasons as the MRB: it can be deployed on its own in some case and it is also possible that MRB and MRF might come from different VNF Providers for the same project.

The MRF VNF representation is illustrated in figure B.7.

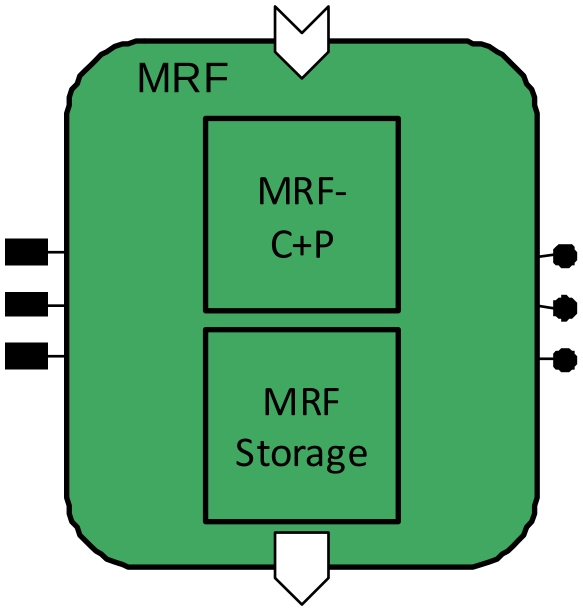

Figure B.7: MRF VNF

There are deployment requirements for the overall MRF VNF and for the MRF-C+P VNFC and for the MRF Storage VNFC, as each component has different needs and reliability model. However as MRF is the VNF, those requirements will all be part of the same VNF descriptor.

Deployment requirements for MRF-C+P:

- The MRF-C+P function is implemented using a N+1 Active redundancy model. In case of MRF-C+P VNFC failure, established calls handled by the failing instance are lost, but new calls are distributed to remaining Active MRF-C+P VNFC instances.
- VM number: 2 to 16 instances. Maximum upper limit is 32, but 16 is a practical upper limit.
- VNF descriptor would also indicate also the number of vCPU needed and vRAM needed.
- For redundancy, anti-affinity rule to force to deploy at least 2 VNFC instances on 2 VMs located on different physical hardware resources.
- Anti-affinity rule to spread MRF-C+P VNFC instances over maximum number of hardware to limit failure impact.
- MRF-P is processing multimedia samples periodically in less than 10 ms. Moving to a virtualised environment, the Digital Signal Processing is moving from specialized HW to software DSP paradigm, it requires specific processing capacity while defining the underlying virtualised infrastructure or evolving that capacity dynamically. While this is an important requirement, it is not clear how this would translate in a VNF descriptor.

Deployment requirements for MRF Storage:

- The MRF Storage function is implemented using a master/slave redundancy model with 1 master VNFC and 1 or more slaves.
- VM number: 2 or more instances.
- VNF descriptor would also indicate also the number of CPU needed and memory needed.
- For redundancy, anti-affinity rule to force to deploy master and slave instances on different VMs located on different physical hardware resources.

Deployment requirements for the MRF VNF:

- Affinity rule to group MRF-C+P VNFCs and MRF Storage VNFCs together on same hardware to get better performance and limit the impact in case of failure.

Operational requirements for the MRF VNF:

Support of typical processes in support of the VNF state transition instantiate, configure, start, stop and terminate.

Relationships:

MRF has 1internal relationships and 2 external ones:

- Relation MRF-C+P - MRF Storage (internal).
- Relation MRF-C+P - MRB.
- Relation MRF-C+P - Application Server. MRF-C has a dependency on an Application Server to be present in the overall solution, but it is not part of the MRF VNF and it might not even by a virtualised component. It would still be useful for NFVO to have an indication of this external dependency in the VNF descriptor.

For all those relations, the requirement is a good network connection with similar characteristics as for MRB relationships: highly Available IP network to guarantee 4 nines and more: no more than one Ping lost in case of network failure, packet Loss less than 0,5 %, round trip delay less than 120 ms, jitter delay less than 40 ms).

B.1.4.3 IMS MRF Deployment

On-boarding and deployment of an IMS MRF service utilizing the MRB and MRF VNFs together with a VNF Forwarding Graph is described in clause A.1 of the MANO GS [i.3].

B.1.5 Standardization Gap

Deployment and operation of a virtualised MRF requires an orchestrator, with open interface.

B.2 DPI Engine VNFC Use Case

The term "DPI" employed here does NOT refer to any of a 3GPP TDF, PCEF, Traffic shaper, etc. product. It is referring to the element that composes such product (and others). Therefore, this clause refers as a "DPI engine".

Most of the L4-L7 networking equipment on the market is using "DPI engine" or "service classification engine".

Such service router, ng-Firewall, GGSN, PCEF, ADC/Load Balancers, Network Analytics Probes, WAN acceleration, BRAS, etc. integrate DPI engine though with different granularity, from the Flow detection, the flow classification, to the Application/Service/network behaviour awareness.

It would be therefore really interesting to look at the DPI engine as a VNFC (VNF Component), which communicates between the different entities as described below.

Though it is NOT an exhaustive list, below are some of the service functions that a DPI engine can deliver howsoever its design:

- Flow detection.
- Flow classification.
- Protocol compliance.
- N-tuple matching.
- Statistical analysis.
- Metadata extraction.
- Heuristic analysis.
- Pattern matching.
- Protocol Signature.
- Malware detection.

Such above DPI engine functions may serve a (virtual) traffic shaper product in enabling the following actions as an illustrative example:

- Bandwidth Limitation.
- Countering/Metering.
- Redirection.
- Gating.
- Application Based Charging.

The APIs below are examples of vendor-defined solutions for implementing the SWA-2 interface interconnecting the VNFCs of a VNF.

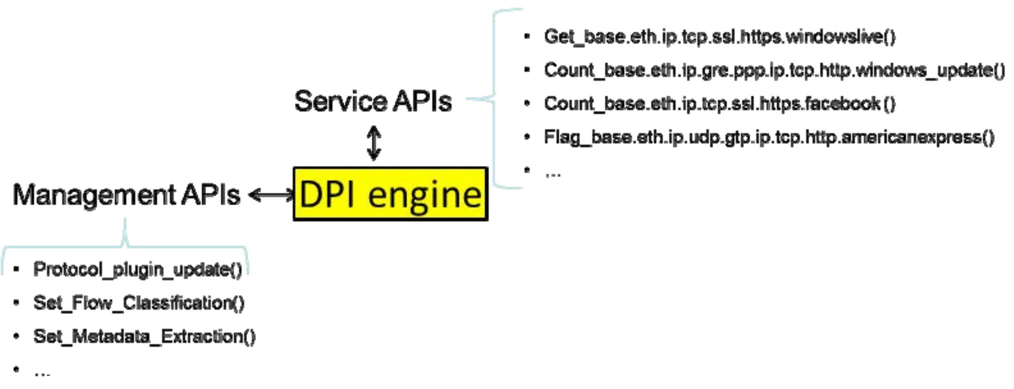
Service APIs
DPI engine
Management APIs
Get_base.eth.ip.tcp.ssl.https.windowslive()
Count_base.eth.ip.gre.ppp.ip.tcp.http.windows_update()
Count_base.eth.ip.tcp.ssl.https.facebook ()
Flag_base.eth.ip.udp.gtp.ip.tcp.http.americanexpress()
...
Protocol_plugin_update()
Set_Flow_Classification()
Set_Metadata_Extraction()
...

Figure B.8: Architecture example of a DPI engine as VNFC

B.2.1 Declination of the DPI Engine

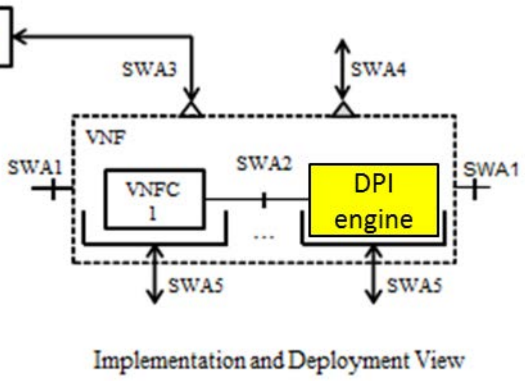

Figure B.9: Implementation and Deployment View

As a reminder, VNF internal interfaces/(V)NFC - (V)NFC interfaces: They are VNF Provider defined, proprietary interfaces interconnecting the (V)NFCs of a (V)NF. Those interfaces typically have performance (capacity, latency, etc.) requirements on the underlying network transport, but are completely opaque from the perspective of an VNF user.

This is indeed the nature and definition of the DPI engine where "interfaces" are used and integrated following specific VNF Provider perspectives.

As agreed with the definition of a VNFC, the DPI engine VNFC does NOT provide neither external interfaces nor external APIs.

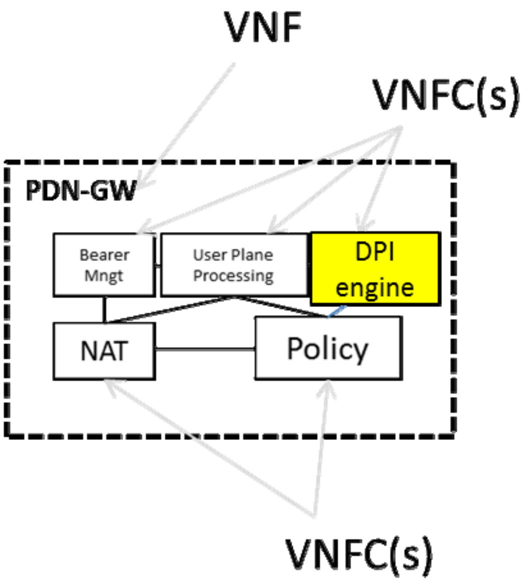

Figure B.10: Example of VNF structure over VNFC(s)

Taking the example of a PDN-GW, it is composed in several VNFC, including the DPI engine.

The DPI engine VNFC does not present any external interfaces and just provide the function of service classification internally. Its implementation follows VNF Provider specific design. VNF Providers may structure and package their software and those components into one or more VM Images, the benefits from having such VNFC characteristics for the DPI engine are multiple (e.g. scalability).

B.2.2 Scalability Benefits

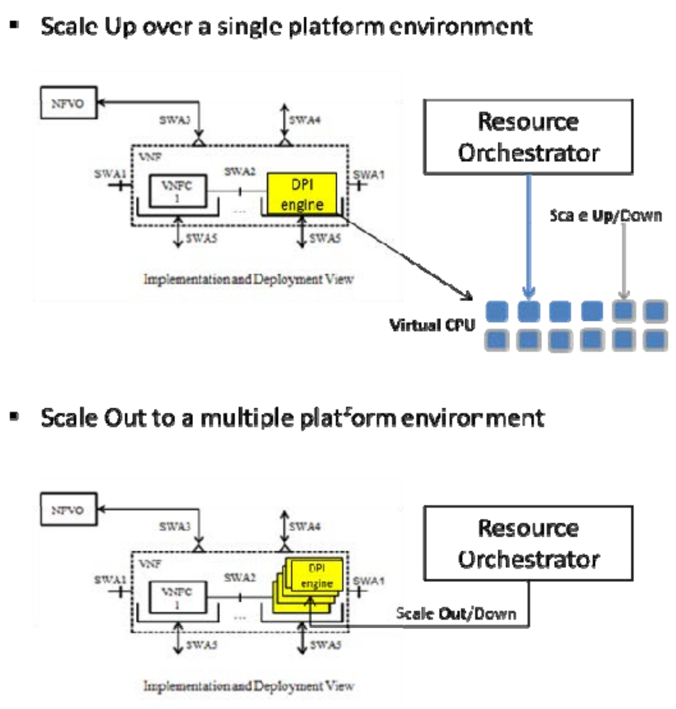

Figure B.11: Example Scalability

- **Scale Up:** By being able to allocate more or less virtual CPU to the DPI engine VNFC, the network management can scale up & down the DPI engine to follow up with the throughput & performance expectations over a single system or a single platform. The DPI engine VNFC through its parallel processing capabilities can benefit from using multiple virtual CPU to handle more service classification sessions.
- **Scale Out:** By being able to multiply the DPI engine VNFC instances, the network management can scale out & down the DPI engine over multiple platform and therefore reach out to the through, performance and architecture specifications.

B.2.3 Security and OpEx Benefits

The second main benefit from having such VNFC characteristics for the DPI engine is the ability to secure and simplify protocol/signature upgrades.

New applications and new existing application specifications require the DPI VNF Providers to maintain constantly an up-to-date protocol & signature database to be integrated into the networking equipment.

Typical characteristic of VNFCs is that their functionality and interfaces change over time as part of the continuous innovation process. For example, in DPI, obfuscated and encrypted protocols are increasingly appearing, and advanced techniques to classify the protocols and signatures are needed. Heuristics, network behaviour and context-aware methodologies are being developed and integrated all the time.

Furthermore, it shall be possible to be integrate new protocol plugins on the fly without interrupting the VNF service. Having such flexibility through the DPI engine VNFC improves and secures the processing of protocol/signatures upgrades.

VNFC upgrade does NOT impact directly the VNF itself.

B.3 Virtual Enterprise Gateway Use Case

Service Introduction

An Enterprise Gateway performs many different types of network functions, such as traffic filtering/policing, traffic monitoring/logging, NAT, IDPS (intrusion detection and prevention systems) and some application-level functions, such as DPI functions. Most enterprises deploy ARs or similar solutions to protect the internal IT infrastructure and enforce policies on the traffic.

While services and traffic types become more diverse and complex, the IT operator may need to design new traffic flow topologies to traverse network devices. The (re)configuration of different types of network functions offered by heterogeneous network devices is time consuming and error prone.

Cloud technology can help to move these boxes into a uniform cloud computing platform. The IT manager may select to buy virtual Enterprise Gateway services provided by network operators, especially for some branches or temporary office sites. The virtual Enterprise Gateway service consists of several network functions that are coupled only at the traffic level. Operators that provide such services can provide a programmable API to users to define their own network and service topology.

Applying the NFV model to physical network devices

A physical network device, such as a network firewall, is illustrated in figure B.12.

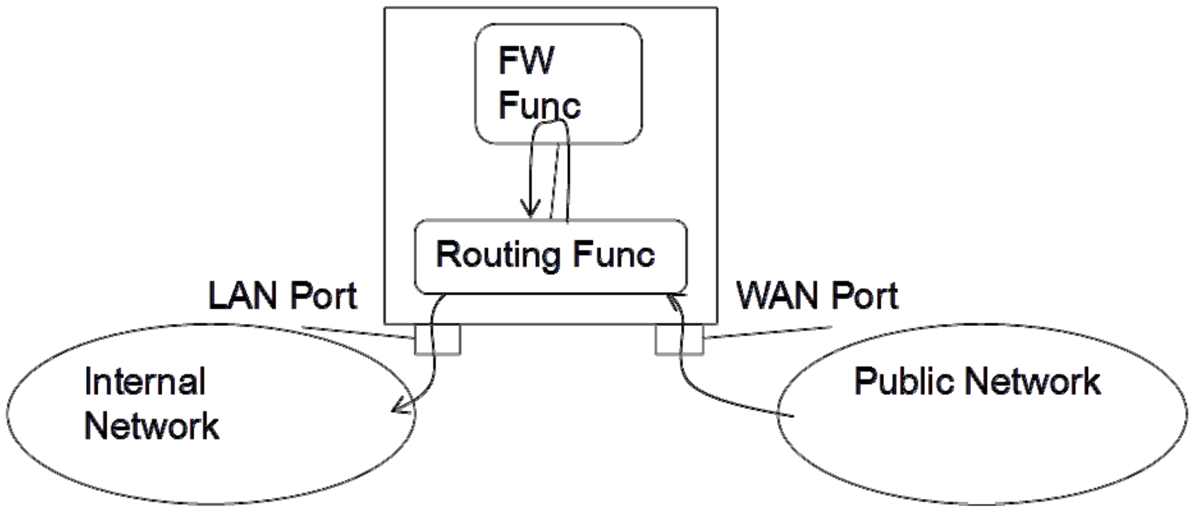

Figure B.12: Physical firewall device model

Figure B.12 shows the concept of a Port. One example of the semantics of a port is provided in Addendum GB922-5PR of the Shared Information and Data (SID) Model, which defines a PhysicalPort as "an actual or potential end point of a topological (physical) link". An example of a physical port is a hardware jack. PhysicalPorts are always contained by another physical object - they can not exist by themselves. The two most common examples are PhysicalPorts on a Card and on a Chassis.

It is important to note that a physical port is not directly programmable. Addendum GB022-5LR of the SID defines a DeviceInterface as an entity that is used to "program a logical connection from a device to a network medium". Different types of DeviceInterfaces exist for the different types of network media (e.g. IP vs. ATM) that are used in a network to enable such media to be programmed. A PhysicalPort may host one or more DeviceInterfaces.

Note that the concept of a port is not the same as an interface. A software interface enables two entities to interact with each other. Software interfaces separate the specification of how software entities communicate from the implementation of the software. They are used to create well-defined points of interaction between software entities, and restrict communication between software entities to those interaction points. Hardware interfaces act in a similar way.

The physical firewall device has at least two ports; one is a WAN port connecting the public network, and the other is a LAN port connecting the internal network. The box has two main functions, a firewall function and a routing function. Theoretically, it is possible that the box does not need any routing functions, and only has a firewall function that receives traffic from one port and sends traffic to another port. However, this design limits the box deployment scenarios, as it cannot handle more than two network segments. Moreover, a set of such designed boxes has to be connected one by one as a node chain. The method of designing service boxes with a routing function enables an operator to design IP links, VRFs and policy routing to enforce service node sequences. The main drawback of this design method is that the service topology is highly coupled with the associated underlying packet network infrastructure.

Cloud environment and technology change the situation, as they enable new design models that separate the routing functions from the other network functions. The NFV architecture supports three main VNF types, which are shown in figure B.13.

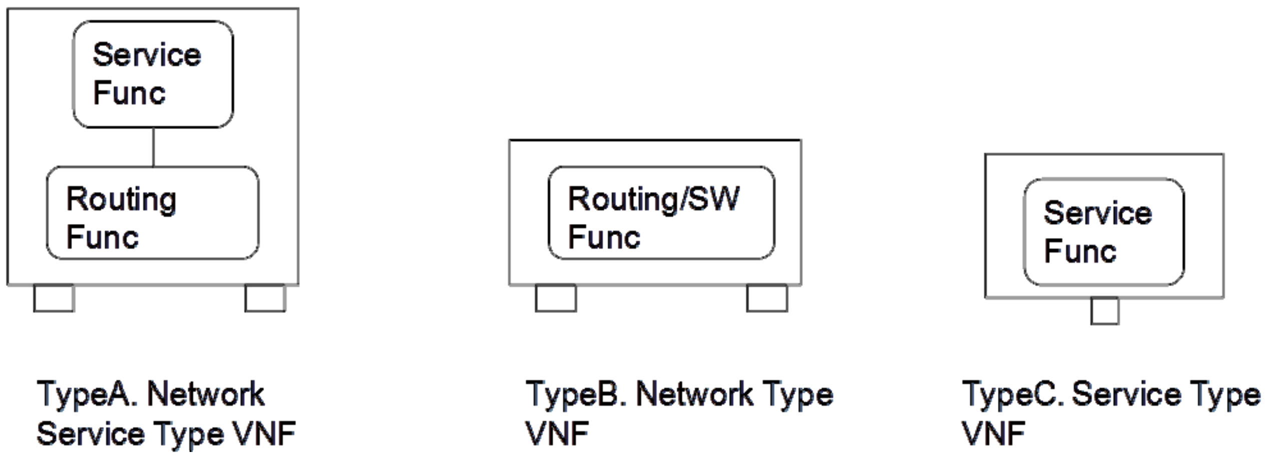

Figure B.13: Three VNF types

In figure B.13, the ports in each type of VNF connect different VNFs to each other. A port may simulate a physical port of a physical network node, but could also be a virtual network.

Considering implementation, one port can map to one virtual port, such as a virtual Ethernet port, on a virtual Network Interface Card (vNIC), or multiple ports can map to one vNIC. Both of these mappings would use a virtualisation container.

Type A is based on virtualising the physical network device without any changes, and hence provides both a routing and a service function. This type of VNF can be managed and operated similarly to the physical network device. Type A is the most similar one to a physical network device among the three, and hence similar EM functions may be used to configure, deploy, and manage it in a VNF. Hence, it has the same operational problems encountered in current physical network devices.

Type B provides just the routing and forwarding function, packaged as a virtualised function. Type A and B may have physical ports that handle packets in the same manner as physical network functions (PNFs) (e.g. packets have some identifying characteristic, such as IP and MAC headers, or labels, that are used for forwarding). A network node of type B has forwarding attributes and network topology information (e.g. an interface or sub-interface identifier, which can be represented by a DeviceInterface), along with port type and other attributes, such as speed (which can be represented by a PhysicalPort) in order to identify its location in a network topology. VNFs that only perform network forwarding and/or routing belong to type B. For example, a virtual switch VNF and a virtual router VNF are VNFs of type B.

Type C is a new type enabled by NFV. It models the service node as a stand-alone VNF without any routing (or forwarding) network function. A type C service node can receive network packets, even if the IP or MAC address does not point to it. It can also receive packets without MAC or IP encapsulation. A type C VNF only cares about the service(s) to be applied to the packet. This is especially ideal for middleboxes (e.g. firewalls, NAT, and WAN optimization). A type C VNF, though visible in a network topology, is focused on service handling. Most type C VNFs only need one port to attach to a network, because they do not provide forwarding and/or routing functions; rather, they provide middlebox functionality. Hence, they attach to the network (via a virtual port), process incoming packets, and send the processed packets to the next NF or VNF. For example, VNFs that only handle control plane packets or only care about service processing belong to this type; examples include MME, IMS, SDN controller and standalone SSL off loader.

Type A inherits properties both from type B and type C, which means it has network forwarding and routing functions as well as other service functions. VNFs, which handle data traffic as well as provide control and/or service functions, belong to this type. Examples of type A include functions such as mobile core network SGW/PGW.

Types A, B, and C are elements that can be used in different NFV-based networks. Their topology defines the functions that they provide to the network. In addition, types A and C may also be used as elements of a network forwarding path.

Virtual Enterprise Gateway service example

The following Virtual Enterprise Gateway example includes a NAT function (type A), a network traffic monitor (NTM) function (type C) and a WAN optimization controller (WoC) function (type A). These functions are defined in the topology shown in figure B.14.

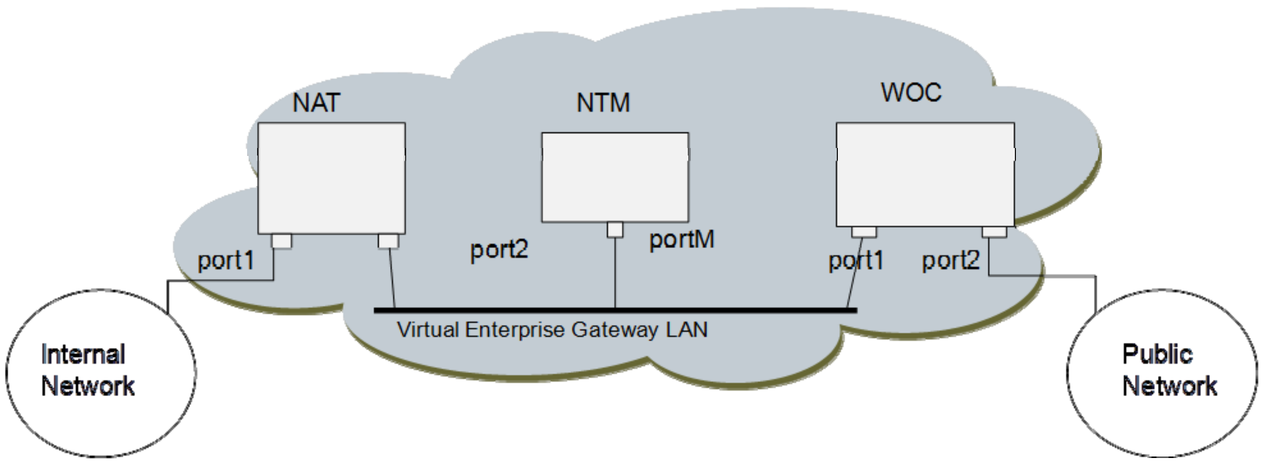

Figure B.14: Virtual Enterprise Gateway network connectivity topology

In figure B.14, there are three VNFs: NAT, NTM, and WOC. NAT and WOC are of Type A, and each has two ports, port1 and port2. NTM is of type C, and only has one port for packet processing. NAT.port2, NTM.portM and WOC.port1 attach to a LAN named 'Virtual Enterprise Gateway LAN'. NAT.port1 connects to the internal network of the enterprise, and WOC.port2 connects to the public network. The network connectivity topology provides a particular service, such as an Ethernet Virtual Service in the form of an ELine, ELAN or ETree (point-to-point, point-to-multipoint, and rooted multipoint, respectively).

For the network topology defined in figure B.14, there is an associated network forwarding path which can be seen in figure B.15 (colored chains).

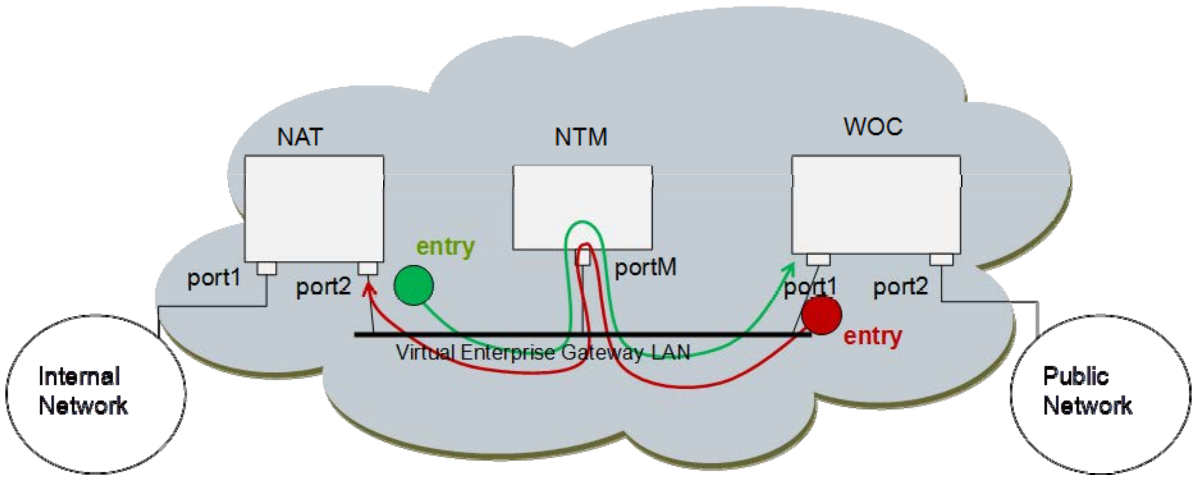

Figure B.15: Virtual Enterprise Gateway network forwarding path

There are two service paths, representing upstream and downstream services, respectively.

Green path:

Entry:	all tcp packets from NAT.port2
Service sequence:	entry--> NTM--> WOC

Red path:

Entry:	all tcp packets from WOC.port1
Service sequence:	entry--> NTM--> NAT

Here, the Virtual Enterprise Gateway LAN provides basic 'LAN' behaviour, receiving packets from each port attached to it and sending the packets to all other ports that are connected to the same LAN. At the same time, the ability to direct any TCP packet through the NTM may also be needed for specific services. This behaviour has higher priority than the normal LAN behaviour for TCP packets.

A network connectivity topology is only concerned with how the different (V)NFs are connected, and how data flows using those connections, regardless of the location and placement of the underlying physical network elements. In contrast, the network forwarding path defines the sequence of (V)NFs to be traversed by a set of packets matching certain criteria. The Network Service contributes to the behaviour of higher layer services, which are characterized by at least performance, dependability, and security specifications, as defined in [i.7], and hence, has different components and description than the network connectivity topology.

The IT operator may want the NTM to redirect some web access to a transparent cache to accelerate content downloading. They may add a new service, such as a VNF Transparent Cache Service, on the Virtual Enterprise Gateway forwarding path. Then, the green path may have a fork in its behaviour, directing content packets tp the VNF Transparent Cache Service, while non-content packets bypass the Transparent Cache Service.

This produces a new Network Connectivity Topology, as well as a new Network Forwarding Path, as shown in figure B.16.

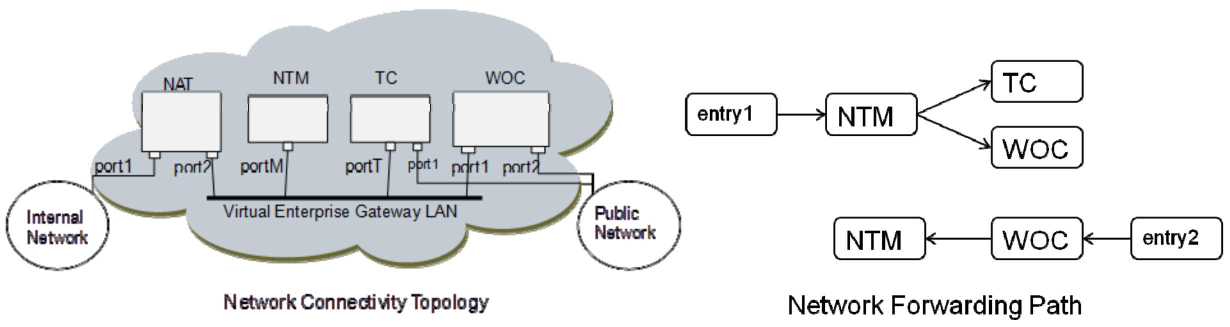

Figure B.16: Network connectivity topology and network forwarding path of the new case

A Network Forwarding Path has components including service nodes with associated service functions, entry and exit points, and an order in which the service functions present at each service node are applied. The order of services applied is application-dependent, and may be a simple sequential set of services, or a more complex graph with alternative paths (e.g. the service may fork, and even later combine), depending on the nature of the traffic, the context of the network, and other factors. A distinguishing characteristic of the Network Forwarding Path is that it specifies such decision-making capabilities; in contrast, a simple forwarding graph has no concept (or even need) of such capabilities.

From the above use case, network nodes and service nodes have different models, and a network topology and the services applied to a network topology may change depending on user requirements, business policies, and/or the context of the network. Moreover, a Network Forwarding Path may be uni-directional or bi-directional, and it may use unicast or broadcast. The definition of a VNF Forwarding Graph addresses only the connectivity requirements of a set of nodes. A Network Forwarding Path thus provides a set of different abstractions than the VNF Forwarding Graph, even though a Network Forwarding Path makes use of the same network connections. Hence, one VNF network connectivity topology may support multiple Network Forwarding Paths, which are under the same provider's administration.

VNF Network Connectivity Topology (VNF-NCT): A graph that defines how its nodes are connected to each other, regardless of the location and placement of the underlying physical network elements. Hence, a VNF-NCT defines a logical topology among (V)NFs in a network. A VNF-NCT is independent of the representation of other aspects of a network, even if they are represented by graphs (e.g. any forwarding graph(s) and/or service graph(s) that are used to represent the forwarding behaviour(s) and service(s) applied to a particular path in the network, respectively). The (logical) topology represented by a VNF-NCT may change as a function of user requirements, business policies, and/or network context.

Examples of VNF-NCTs include:

- E-Line, E-LAN, and E-TREE (defined by the Metro Ethernet Forum in MEF Technical Specification MEF 6.1: Ethernet Services Definitions - Phase 2", April, 2008).
- VPLS and VPWS Services (e.g. defined by IETF RFC 4761 [i.32]).
- Different types of Virtual LANs or Private Virtual LANs (e.g. IETF RFC 3069 [i.28]).
- Different types of Layer 2 Virtual Private Networks (e.g. IETF RFC 4464 [i.31]).
- Different types of Layer 3 Virtual Private Networks (e.g. IETF RFC 3809 [i.29]).
- Different types of Multi-Protocol Label Switching Networks (e.g. IETF RFC 3031 [i.27]).
- Other types of layer 2 services, such as Pseudowire Switching for providing multiple Virtual Leased Line Services (e.g. IETF RFC 4385 [i.30]).

A VNF Service Graph (VNF-SG): A graph specified by a Network Service Provider that defines how one or more network services are applied to traffic flowing through a set of nodes of the graph. A VNF-SG is a directed acyclic graph, where at least one node is a VNF through which network service traffic is directed. A VNF-SG may contain VNF Sets as well as nested VNF-SGs. A VNF-SG is independent of the representation of other aspects of a network, even if they are represented by graphs (e.g. any forwarding graph(s) and/or VNF-NCTs that are used to represent the forwarding behaviour(s) and logical topology applied to a particular path in the network, respectively). The services applied to a particular type of traffic flowing through a VNF-SG may change as a function of user requirements, business policies, and/or network context.

The difference between a VNF-FG, a VNF-NCT, and a VNF-SG is illustrated in figure B.17.

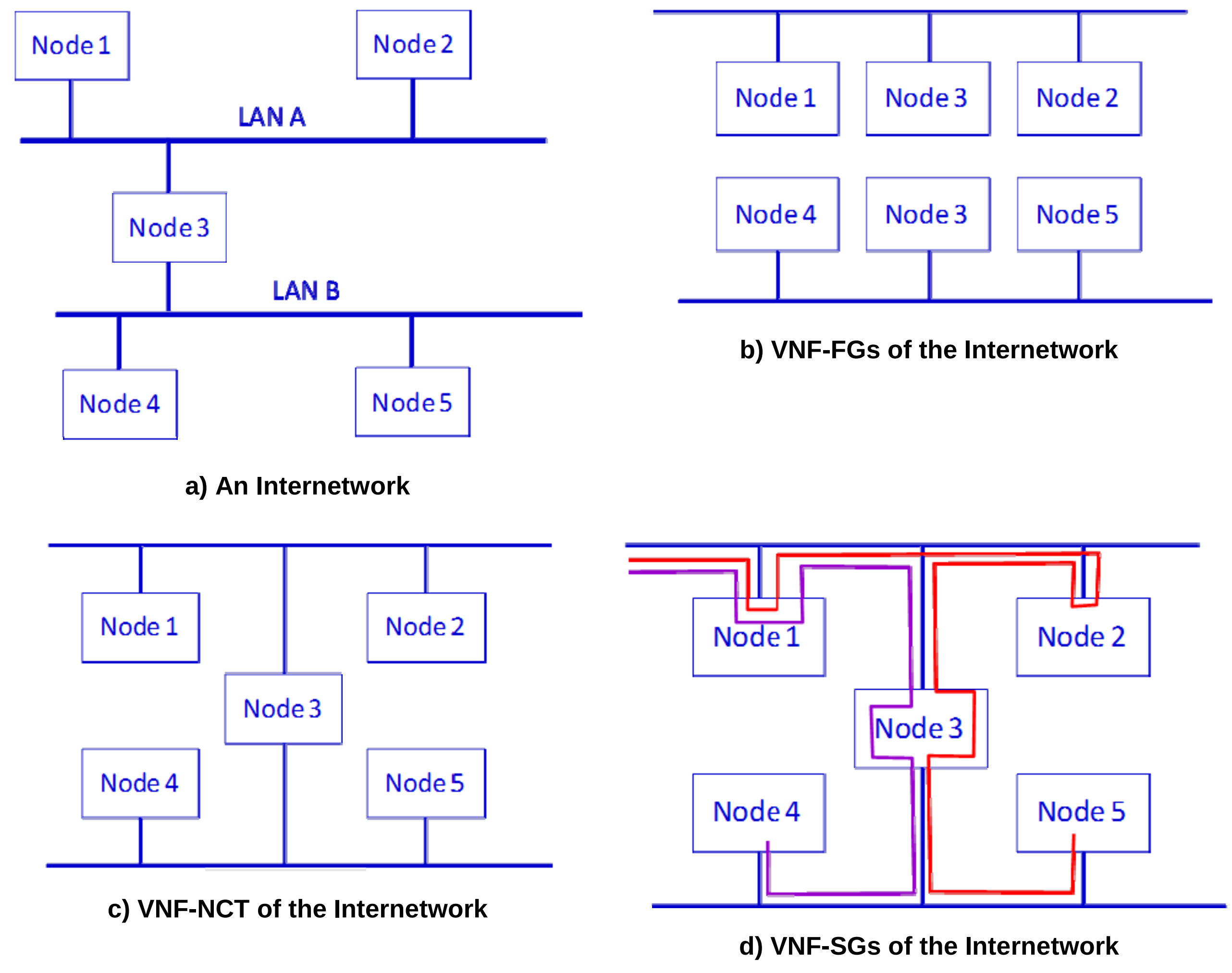

Figure B.17: VNF-FG, VNF-NCT and VNF-SG

Figure B.17 a) shows a simple internetwork: two different LANs are interconnected by a border router (node 3). Figure B.17 b) shows the corresponding forwarding graphs. Figure B.17 c) shows the corresponding NCT. Note that the NCT shows both routing and forwarding, whereas the VNF-FG shows only two separated LANs. Figure B.17 d) shows an exemplar set of services produced by using (for example) policy-based routing. In this example, the purple traffic is sent directly from LAN A (node 1) to LAN B (node 4) through node 3. However, the red traffic adds the services provided by node 2, and routes to a different node (node 5) in LAN B. A simple example of this could be filtering on a ToS or DSCP byte, or routing based on source addresses, or a number of other different applications.

B.4 TDF as VNF Use Case

This clause is aimed to describe a TDF (Traffic Detection Function - 3GPP) use-case instantiated as a virtualised Network Function VNF. This example is particularly relevant as it is part of EPC (Evolved packet Core), listed as one of top priorities for operators to be virtualised. The TDF use case is also interesting because it combines signalling and user/data plane functions. This use case description is intended to help identifying potential issues and standardization gaps for virtualising 3GPP vEPC network functions and communicate them to 3GPP.

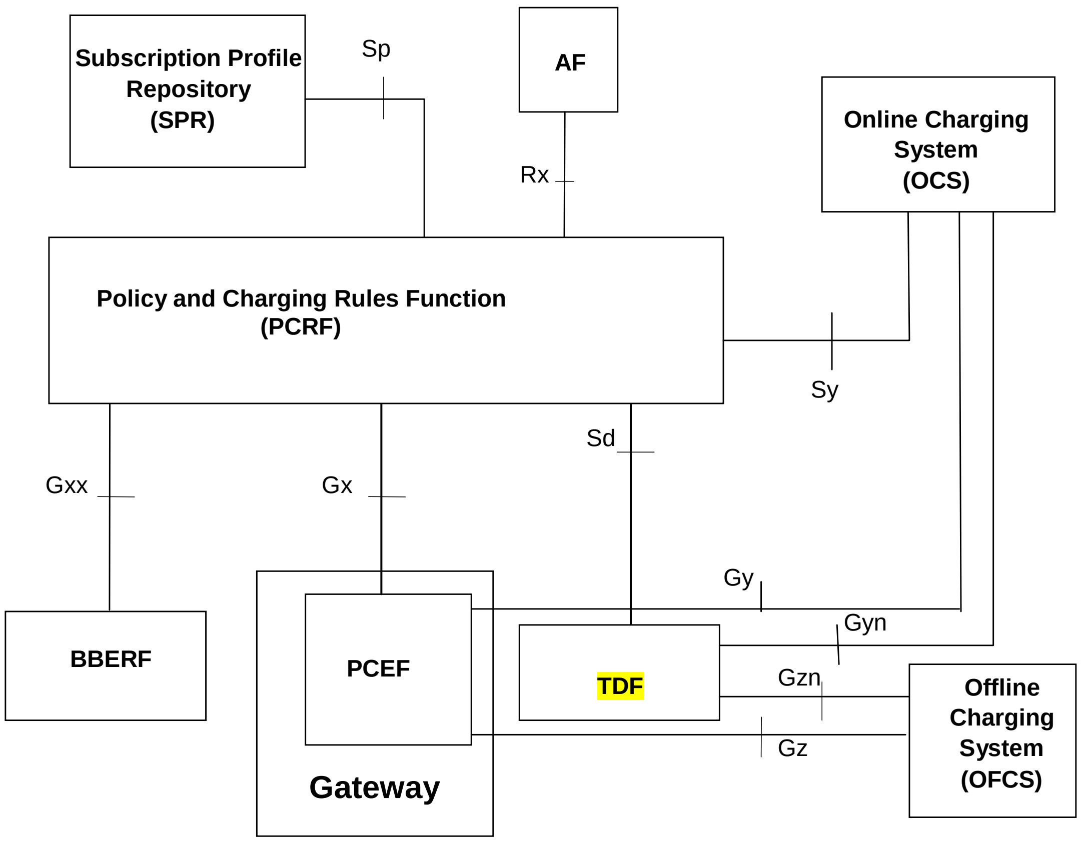
Subscription Profile Repository (SPR)
Sp
AF
Rx
Online Charging System (OCS)
Policy and Charging Rules Function (PCRF)
Sy
Sd
Gxx
Gx
Gy
Gyn
BBERF
PCEF
TDF
Gzn
Offline Charging System (OFCS)
Gz
Gateway

Figure B.18: Overall PCC logical architecture (non-roaming)

As per ETSI TS 123 203 [i.15], the TDF is a functional entity that performs application detection by applying packet inspection (DPI) mechanisms, reporting of detected application to the PCRF and detected application's enforcement (gating, bandwidth limitation, redirection), usage monitoring report to the PCRF and charging report to the OCS and/or OFCS. The similar functionality can also be implemented by P-GW according to ETSI TS 123 203 [i.15], in case of PCEF enhanced with Application Detection and Control (ADC) feature.

B.4.1 Functional Block Description

Figure B.19 depicts an example where a TDF instance is decomposed into 4 main functional blocks.

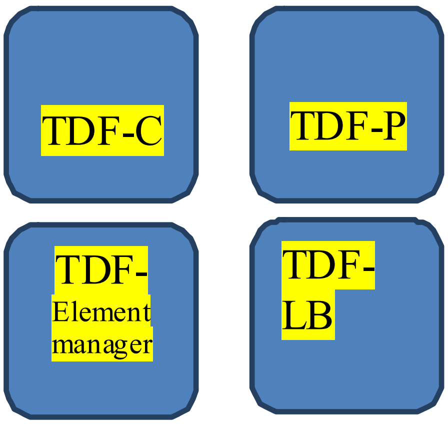

Figure B.19: TDF instantiation in a virtualised environment

The **TDF-C (Control) Functional block** represents the controlling function of the TDF; it is a signalling function that terminates the Sd, Gyn and Gzn Control interfaces and also Orchestration and management control interface and drives the TDF-P functional entity with regards to the required functionality, including enforcement and charging parameters control per each one of the required applications as per Application Detection and Control (ADC) Rules received from the PCRF and charging related requirements, received from the OCS. The TDF-C can also provide features like off-line reporting of TDF operations. The similar controlling function can be implemented by the P-GW as a part of P-GW Control block in case f PCEF enhanced with ADC with the difference that there are Gx, Gy and Gz Control interfaces then.

The **TDF-P (Processing) Functional block** represents the processing function of the TDF; it connects to the Gi/SGi data interface that is used to carry user plane traffic and performs TDF session's classification based on packet inspection (either shallow packet inspection or deep packet inspection (DPI))/processes the traffic/implements the TDF VNF Network Forwarding Graph while TDF VNF acts as a Service Chain Classification Node following the rules received from the TDF-C Functional Block. It is a data plane function. One example implementation could be assimilated to the DPI Engine VNFC Use Case clause B.2 description as long as it does not require packet processing functions as described in 3GPP specifications. The similar processing function can be implemented by the PGW as a part of P-GW Processing block in case f PCEF enhanced with ADC with the difference that P-GW originates/terminates GTP or PMIP interface.

The **TDF-LB (Load Balancer) Functional block** is aimed to load balance the incoming signalling towards several TDF-C instances.

The **TDF Element Manager Functional block** is in charge of:

- (Specific functionality for TDF functional operation): Manage the interactions between these functional blocks in terms of e.g. controlling TDF-P functionality based on external Application detection and Control (ADC) Rules received from the PCRF.

- (Specific functionality for TDF FCAPS): Request and Collect metrics from outside system and compute, e.g. amount of Diameter messages per Second per each one of Diameter interfaces per each TDF-C instance, time to establish cellular TDF session, time to fetch Resources, build network analytics based on this information and also on TDF session information (e.g. location access network information, subscriber information etc.) received from TDF-C functional block as per operator's requirements, etc.

Note that mapping the TDF-C and TDF-P on two Functional Blocks for instance is only functional; the mapping to the virtual deployment can lead the TDF-C and TDF-P to be grouped together. One of the reasons is that functionality of TDF-C depends on User-Pane data received and processed by TDF-P Functional block. TDF-C block can also be instantiated further e.g. to Block which handles signalling interactions with PCRF and block handling signalling interactions with OCS/OFCS.

B.4.2 TDF Functional Blocks within the NFV architecture

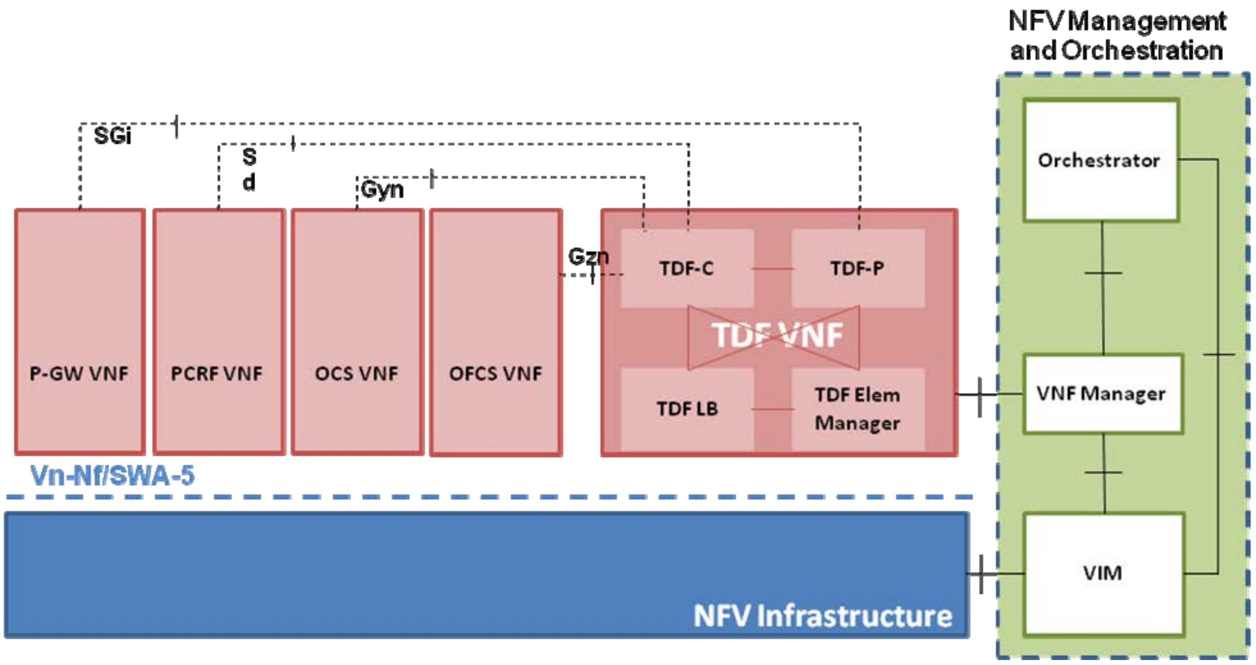

Figure B.20: TDF Functional Blocks within the NFV architecture

B.4.3 Existing interfaces (as per ETSI TS 123 203)

This clause is provided for general description of the existing 3GPP specification (ETSI TS 123 203 [i.15]).

SGi

The SGi is a Reference point between the P-GW and the PDN (Packet Data Network). TDF resides on this interface. From the vEPC perspective, TDF VNF communicates with P-GW VNF for User-Plane data transfer in both directions.

Sd

The Sd reference point resides between the PCRF and the TDF.

The Sd reference point enables a PCRF to have dynamic control over the application detection, enforcement and charging behaviour at a TDF.

The Sd interface is specified in the ETSI TS 123 203 [i.15].

Gyn

The Gyn reference point resides between the OCS and the TDF.

The Gyn reference point allows online credit control for charging. The functionalities required across the Gyn reference point are defined in the ETSI TS 132 251 [i.16].

Gzn

The Gzn reference point resides between the TDF and the OFCS.

The Gzn reference point enables transport of offline charging information.

The Gzn interface is specified in the ETSI TS 132 240 [i.17].

NOTE: All the interfaces described above are part of the existing 3GPP architecture.

Annex C (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Thinh Nguyenphu, Nokia Networks

Other contributors:

Srini Addepalli, Freescale
Nicolas Bouthors, Qosmos
Michael Brenner, Alcatel-Lucent
Michael K Bugenhagen, Centurylink
Mike Bursell, Citrix
Jean-Marie Calmel, Oracle
Jorge Carapinha, PT SGPS
Bruno Chatras, Orange
Sachin Chickballapur, Juniper Networks
Dan Druta, AT&T
Farid Feisullin, Sprint
Marc Flauw, HP
António Gamelas, PT SGPS
Alla Goldner, Allot Communications
Kalyanjeet Gogoi, Juniper Networks
Bin Hu, AT&T
Jie Hu, ZTE
Yong Huang (Oliver), Huawei
Fred Huve, HP
Ashiq Khan, NTT DOCOMO
Uli Kleber, Huawei
Hongyu Li, Huawei
Tommy Lindgren, Ericsson
Shucheng Liu (Will), Huawei
Astrid Mann, Huawei
Arturo Martin De Nicolas,Ericsson
Bharat K. Mota, Freescale
Tetsuya Nakamura, NTT DOCOMO
Karl-Heinz Nenner, Deutsche Telekom
Marie-Paule Odini, HP
Neal Oliver, Intel
Magnus Olsson, Ericsson
François-Frédéric Ozog, 6WIND
Uwe Rauschenbach, Nokia Networks
Ron Sidi, Contextream
Bertrand Souville, DOCOMO Communications Lab
John Strassner, Huawei
Joan Triay, DOCOMO Communications Lab
Thomas Wocalewski, Huawei
Steven Wright, AT&T
Amanda Xiang, Huawei
Parviz Yegani, Juniper Networks
Frank Zdarsky, NEC

Annex D (informative): Bibliography

- ETSI GS NFV-INF 002: "Network Functions Virtualisation (NFV); Infrastructure; Illustrative Use Cases".
- ETSI GS NFV-INF 003: "Network Functions Virtualisation (NFV); Infrastructure; Compute Domain".
- ETSI GS NFV-INF 004: "Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain".
- ETSI GS NFV-INF 007: "Network Functions Virtualisation (NFV); Infrastructure; Methodology to describe Interfaces and Abstractions".
- ETSI GS NFV-INF 008: "Network Functions Virtualisation (NFV); Infrastructure; Portability and Replicability".
- Recommendation ITU-T Y.2011 (10/2004): "General principles and general reference model for Next Generation Networks".

History

Document history		
V1.1.1	December 2014	Publication