ETSI GS NFV-PER 001 V1.1.1 (2014-06)

Network Functions Virtualisation (NFV);
NFV Performance & Portability Best Practises

Disclaimer

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

Reference
DGS/NFV-PER001

Keywords
performance, portability

ETSI

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

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

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

Contents

Intellectual Property Rights

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

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

Foreword

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

Modal verbs terminology

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

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

1 Scope

The present document provides a list of features which the performance and portability templates (Virtual Machine Descriptor and Compute Host Descriptor) should contain for the appropriate deployment of Virtual Machines over a Compute Host (i.e. a "telco datacentre").

In addition, the document provides a set of recommendations and best practises on the minimum requirements that the HW and hypervisor should have for a "telco datacentre" suitable for data-plane workloads. The recommendations and best practises are based on tests results from the performance evaluation of data-plane workloads. It is recognized that the recommendations are required for VNFs supporting data plane workloads and that a small portion of the recommended list are not required in all cases of VNFs, such as VNFs related to control plane workloads.

2 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.

2.1 Normative references

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

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

[2] ETSI GS NFV 001: "Network Functions Virtualisation (NFV); Use Cases".

2.2 Informative references

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] Open Virtualisation Format Specification Version 2.1.0.

NOTE: Available at: http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.0.pdf.

[i.2] Libvirt - The Virtualisation API.

NOTE: Available at: http://libvirt.org/.

[i.3] AWS® CloudFormation.

NOTE 1: Available at: http://aws.amazon.com/cloudformation/.

NOTE 2: AWS® CloudFormation 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.

[i.4] Portable Hardware Locality.

NOTE: Available at: http://www.open-mpi.org/projects/hwloc/.

[i.5] IETF RFC 2544: "Benchmarking Methodology for Network Interconnect Devices".

NOTE: Available at: http://www.ietf.org/rfc/rfc2544.txt.

[i.6] IETF RFC 2889: "Benchmarking Methodology for LAN Switching Devices".

NOTE: Available at: http://www.ietf.org/rfc/rfc2889.txt.

[i.7] IETF RFC 3918: "Methodology for IP Multicast Benchmarking".

NOTE: Available at: http://www.ietf.org/rfc/rfc3918.txt.

[i.8] IETF RFC 3511: "Benchmarking Methodology for Firewall Performance".

NOTE: Available at: http://tools.ietf.org/rfc/rfc3511.txt.

[i.9] IEEE 1588: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document the terms and definitions given in GS NFV 003 [1] and the following apply:

NOTE: A term defined in the present document takes precedence over the definition of the same term, if any, in GS NFV 003 [1].

Network Function (NF): A functional building block within a network infrastructure, which has well-defined external interfaces and a well-defined functional behaviour. In practical terms, a Network Function is today often a network node or physical appliance.

Network Functions Virtualisation Infrastructure (NFVI): The NFV-Infrastructure is the totality of all hardware and software components which build up the environment in which VNFs are deployed. The NFV-Infrastructure can span across several locations, i.e. multiple N-PoPs. The network providing connectivity between these locations is regarded to be part of the NFV-Infrastructure. The NFV-Infrastructure includes resources for computation, networking and storage.

Compute Host: A Compute Host is the whole server entity, part of an NFVI, composed of a HW platform (processor, memory, I/O devices, internal disk) and a hypervisor running on it.

Compute Host Descriptor (CHD): A Compute Host Descriptor is a template to define a storage schema for the capabilities and up-to-date available resources which can be offered by a Compute Host server to VM Images at deployment time. Therefore, there will be one Compute Host Descriptor for every Compute Host, containing both its capabilities and its available resources.

Virtualised Network Function (VNF): An implementation of an NF that can be deployed on a Network Function Virtualisation Infrastructure (NFVI). A VNF can be deployed as a set of Virtual Machines (VM), as SW components deployable, maintainable and manageable, emulating a single computer.

Virtual Machine Instance (VM Instance): A Virtual Machine Instance is a Virtual Machine already running in a Compute Host.

Virtual Machine Image (VM Image): A VM Image is an executable SW component, subject to be deployed in Compute Hosts as one or several Virtual Machine Instances.

Virtual Machine Configuration (VM Configuration): A VM Configuration is the final configuration to be applied when deploying a VM Image on a specific Compute Host.

3.2 Abbreviations

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

AAA	Authentication, Authorization, and Accounting
API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Application Server
BBU	Base Band Unit
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
BIOS	Basic Input/Output System
BNG	Broadband Network Gateway
BRAS	Broadband Remote Access Server
BW	Bandwidth
CDN	Content Delivery Network
CGNAT	Carrier Grade Network Address Translation
CHD	Compute Host Descriptor
CIFS	Common Internet File System
COTS	Commercial Off-The-Shelf
CPE	Customer Premises Equipment
CPU	Central Processing Unit
C-RAN	Cloud-Radio Access Network
CVLAN	Customer VLAN
DCB	Data Center Bridging
DDoS	Distributed Denial of Service
DDR2	Double Data Rate type 2
DDR3	Double Data Rate type 3
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DPI	Deep Packet Inspection
DSLAM	Digital Subscriber Line Access Multiplexer
DUT	Device Under Test
E-CPE	Enterprise-Customer Premises Equipment
ERPS	Ethernet Ring Protection Switching
FFT	Fast Fourier Transform
FIB	Forwarding Information Base
FTP	File Transfer Protocol
FW	Firewall
GB	GigaByte
GE	Gigabit Ethernet
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GPS	Global Positioning System
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
GW	Gateway
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HW	Hardware
I/O	Input/Output
I-CSCF	Interrogating-Call Session Control Function
IMS	IP Multimedia Subsystem
IO	Input Output
IOMMU	Input/Output Memory Management Unit
IOTLB	I/O Translation Lookaside Buffer
IP	Internet Protocol
IPC	Inter-Process Communication
IPoE	IP over Ethernet
IPsec	IP security
ISIS	Intermediate System to Intermediate System

KPI	Key Performance Indicator
L4	Layer 4
L7	Layer 7
LPM	Longest Prefix Match
MAC	Media Access Control
MAN	Metropolitan Area Network
MANO	MANagement and Orchestration
MGCF	Media Gateway Controller Function
MME	Mobility Management Entity
MMU	Memory Management Unit
MOS	Mean Opinion Score
MOS-AV	MOS-Audio & Video
MPLS	Multi-Protocol Label Switching
MSE	Mean Square Error
N/A	Not Applicable
NAS	Network-Attached Storage
NAT	Network Address Translation
NF	Network Function
NFV	Network Functions Virtualisation
NFVI	Network Functions Virtualisation Infrastructure
NIC	Network Interface Card
NUMA	Non-Uniform Memory Access
OLT	Optical Line Terminal
ONT	Optical Network Terminal
ONU	Optical Network Unit
OS	Operating System
OSPF	Open Shortest Path First
OVF	Open Virtualisation Format
P2P	Peer-to-Peer
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PCIe	VF PCIe Virtual Function
PCI-SIG	PCI Special Interest Group
PCRF	Policy and Charging Rules Function
P-CSCF	Proxy-Call Session Control Function
PDN	Packet Data Network
PE	Provider Edge
P-GW	PDN-Gateway
PNF	Physical Network Function
PPP	Point-to-Point Protocol
PPPoE	Point-to-Point Protocol over Ethernet
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QoS	Quality of Service
R/W	Read/Write
RADIUS	Remote Authentication Dial In User Service
RAM	Random Access Memory
RAN	Radio Access Network
RDMA	Remote Direct Memory Access
RGW	Residential Gateway
RIB	Routing Information Base
RMS	Root Mean of Squares
RoCE	RDMA over Converged Ethernet
RX	Reception
SAN	Storage Area Network
S-CSCF	Serving-Call Session Control Function
SGSN	Serving GPRS Support Node
S-GW	Serving-Gateway
SLA	Service Level Agreement
SMT	Simultaneous Multi-Threading
SR-IOV	Single Root I/O Virtualisation
SSIM	Structural Similarity

STB	Set-Top-Box
STP	Spanning Tree Protocol
SVLAN	Service VLAN
SW	Software
TCP	Transmission Control Protocol
TLB	Translation Lookaside Buffer
TWAMP	Two-Way Active Measurement Protocol
TX	Transmission
VF	Virtual Function
VIA	Virtual Interface Architecture
VIM	Virtualised Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtualised Network Function
VQM	Video Quality Metric
VTA	Virtual Test Appliance
EMS	Element Management System
KVM	Kernel-based Virtual Machine
DNS	Domain Name System
NTP	Network Time Protocol
SSH	Secure SHell
NFS	Network File System
TA	Test Agent
ISV	Independent Software Vendor
ABR	Available Bit Rate
CBR	Constant Bit Rate
VBR	Variable Bit Rate
RSS	Receive Side Scaling

4 Introduction

The present document provides a list of minimal features which the VM Descriptor and Compute Host Descriptor should contain for the appropriate deployment of VM Images over an NFVI (i.e. a "telco datacentre"), in order to guarantee high and predictable performance of data plane workloads while assuring their portability. In addition, the document provides a set of recommendations on the minimum requirements which HW and hypervisor should have for a "telco datacentre" suitable for different workloads (data-plane, control-plane, etc.) present in VNFs.

It should be noted that the present document focuses on capturing HW and SW requirements for VMs dealing with data-plane workloads.

As shown in figure 1, a VNF can be composed of a set of VMs , and their internal and external interfaces.

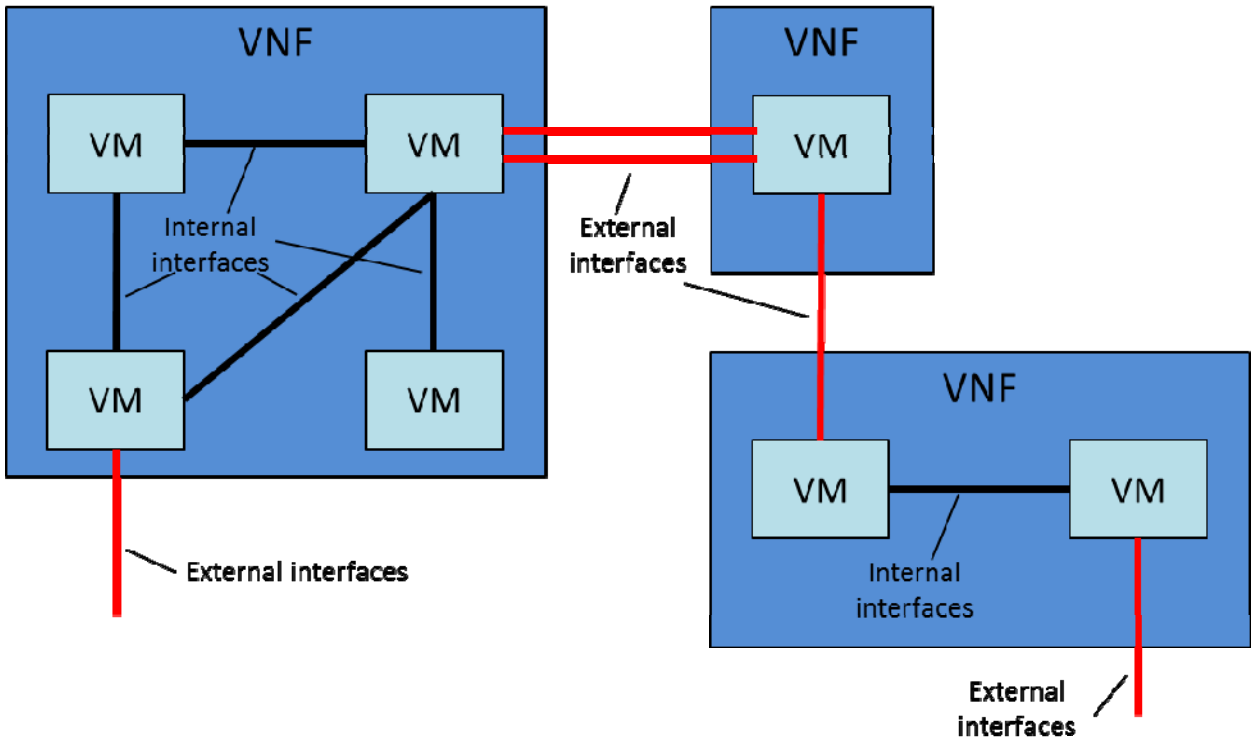


Figure 1: VNF and VMs

Regarding NFV-Infrastructure, the document will focus on NFVI compute domain, leaving storage and networking HW domains for further analysis. Regarding NFV SW, only hypervisor capabilities are considered at this stage. The term "**Compute Host**" will be used to refer to the whole entity composed of the HW platform (processor, memory, I/O devices, internal disk) and a hypervisor running on it. Scenarios with two or more hypervisors on the same server are not considered for this first analysis.

In the present document, the term Compute Host Descriptor (CHD) will be used - applied to every Compute Host - to describe the template defining a storage schema for the capabilities and currently available resources offered by the Compute Host for the deployment of VM Images. Hence, there will be one CHD for every Compute Host, containing both its capabilities and its available resources. In a practical realization, since the capabilities can be the same for a common type of Compute Host (HW and hypervisor), the CHD could be implemented as the combination of two different templates -one containing the capabilities, the other one containing the available resources-, thus being possible to have a common template for the capabilities of a given type of Compute Host. Along the document, the term CHD will sometimes be used to refer to one of its potential practical realizations: a parseable template file describing both the key HW capabilities and available resources of server Compute Host.

On the other hand, for the purposes of performance and portability discussions, it is required a term to describe the NFV resources which are required by a VM Image from the NFVI. Additionally, for a proper discussion on the subject, it is required a term to distinguish the demanded resources from the applied configuration. For this purpose, in the present document the following new terms are coined:

- **Virtual Machine Descriptor** (VM Descriptor) is a template which declares the NFV resources to be required by the VNF's VMs from the NFVI. It is a type of meta-information associated with a VM Image. The description is made in the form of a SW template, which is computable by the MANO Functions. E.g. if OVF [i.1] is used as the way of packaging VM images, then VM Descriptor could be part of the OVF meta-data. It is expected a single VM Descriptor for each VM Image type, independently on the number of VM Instances deployed on an NFVI.

- **Virtual Machine Configuration** (VM Configuration) is the final configuration to be applied when deploying a VM Image on a specific Compute Host. E.g. If libvirt [i.2] was being used by the hypervisor manager, then the VM Configuration could be analogous to the *vm.xml* that is generated at deployment (containing host-specific information such as PCI device ids, core ids, etc.). It is expected a VM configuration for each deployed VM Instance.

Along the document, for the sake of simplicity, the term VM Descriptor might also be used as synonymous of one of its potential practical realizations: a parseable template file describing the requirements of a VM Image. Likewise, several VM Descriptors could be joined together in a "**template file**" (e.g. analogous to AWS® CloudFormation templates [i.3], see note) describing the whole VNF - comprising its components (VMs), the internal interfaces between VMs and the set of external interfaces.

NOTE: AWS® CloudFormation 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.

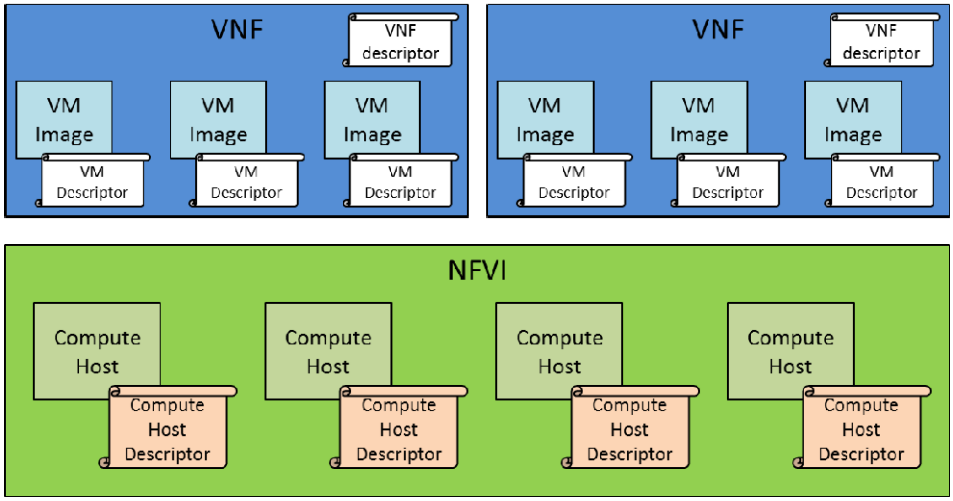


Figure 2: Compute Host Descriptor (CHD) and VM Descriptor templates

We will assume scenarios where VNFs might be provided by VNF SW vendors as SW packages including a descriptor for the whole VNF, a set of VM Images, and their corresponding VM Descriptors. In the simplest scenario of deployment of a VNF, with a VNF composed of a single VM Image, and after all the preparatory steps (e.g. inventory check), the MANO Functions would read the VM Descriptor and parse the requirements for the underlying HW. Next, it would check what Compute Hosts would be suitable to deploy the VM Image, based on the different Compute Host Descriptors. Those CHDs should show up-to-date availability of resources (free CPUs, free memory, etc.). From the set of valid Compute Hosts, the MANO Functions will choose one (the nature of this selection procedure is not relevant for the purpose of the present document), and, then, a VM Configuration will be elaborated, containing the specific resources that should be consumed and configured on the selected Compute Host. That VM Configuration will be used to deploy the VM Image on the selected Compute Host and, therefore, is specific for that server since it may ask for specific devices or resources. Finally, the VM Image is deployed on the server, launched by the MANO Functions.

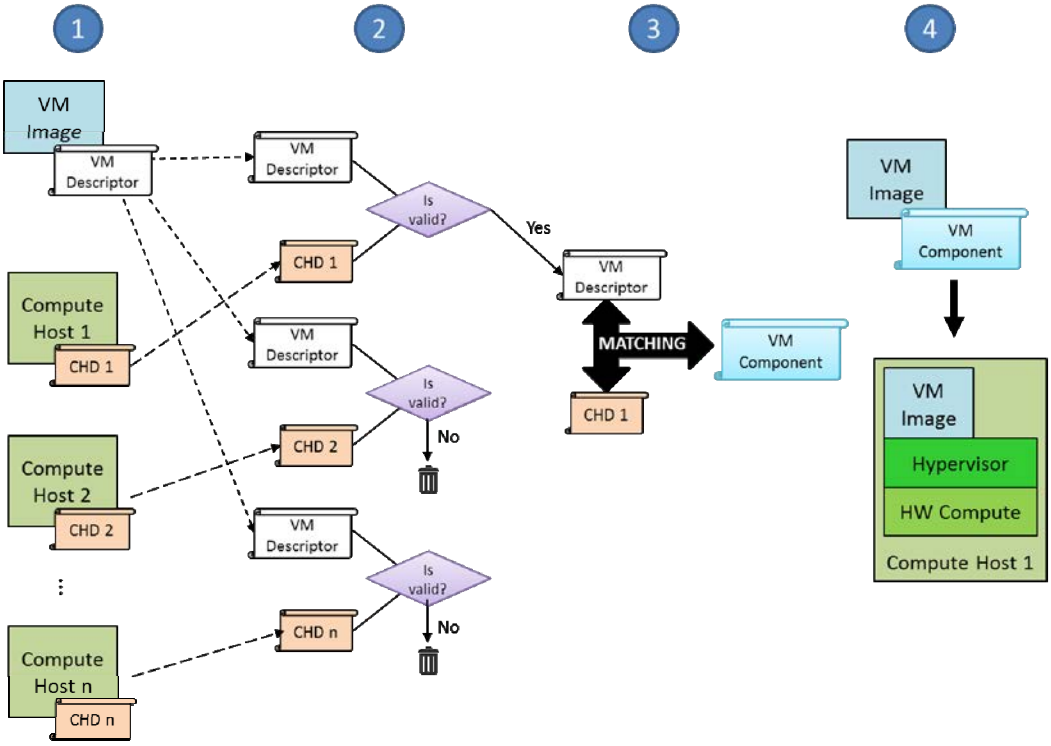


Figure 3: Deployment process as part of the orchestration process

The purpose of the present document is to provide the list of VM requirements that should be included in the VM Descriptor template, and the list of HW capabilities that should be included in the Compute Host Descriptor (CHD) to assure predictable high performance. It is assumed that the MANO Functions will make the mix & match. The details of the orchestration process, the VM Configuration and the specific format of the CHD and VM Descriptor templates are out of the scope of the present document.

The present document is structured as follows. First, clause 5 describes the methodology used to identify the list of requirements and capabilities for the VM Descriptor and CHD templates. The methodology is based on experimentation through tests and comparison of the test results in both bare metal and virtualisation environments in order to identify HW and SW bottlenecks. Instead of testing all the possible VNFs, a pragmatic approach has been followed, focusing on significant VNFs already described in the NFV use cases [2]. Moreover, instead of testing the whole VNF, tests have been restricted to specific tasks or workloads, which happen to be present in several VNFs. Relevant workloads in terms of performance have been identified and are also described in clause 5. The clause also introduces a first collection of tests (completed or planned) covering some of these workloads.

Next, clause 6 presents the main conclusions for each tested workload, highlighting those relevant HW and SW features and how they affect performance. The present document covers specifically the analysis of data plane workloads in intermediate elements.

Based on the workload analysis in clause 6, clause 7 provides a recommendation on the minimum requirements that a Compute Host should have for a "telco datacentre" suitable for data-plane workloads, and, finally, clause 8 gathers the list of features which the Compute Node Descriptor and the VM Descriptor templates should contain for the appropriate deployment of VM Images over an NFVI (i.e. a "telco datacentre"), in order to guarantee high and predictable performance while preserving portability across different servers.

5 Methodology and relation to NFV use cases

Testing all possible VNFs would be an extensive task. The followed approach has been, instead, to run tests for relevant network tasks or workloads, which happen to be present in common VNFs. This strategy simplifies the problem allowing us extending the conclusions from a specific workload in a VNF to that very workload in other VNFs.

For the sake of performance analysis, the following workload types are distinguished:

- **Data plane workloads**, which cover all tasks related to packet handling in an end-to-end communication between edge applications. These tasks are expected to be very intensive in I/O operations and memory R/W operations.
  - In the case of an edge NF such as a CDN cache node, this workload includes the L4-L7 session initiation and termination, and the data transmission and reception. This typically would mean a high load in R/W operations from/to memory and in I/O operations related to data transmission and reception.
  - In the case of an intermediate NF such as a router, the tasks cover removing/adding headers, forwarding packets, modifying packet fields, flow/session accounting, etc. This means a high load in R/W operations from/to memory and in I/O operations related to data forwarding. Due to the relative simplicity of these tasks, in some circumstances it might be feasible to bypass some OS tasks in order to circumvent potential bottlenecks.
  - In the case of an intermediate NF with encryption functions such as an IPSec tunneller, besides the packet handling mentioned above, per-packet encryption becomes a key task, meaning an extra load in CPU computing resources.
- **Control plane workloads**, which cover any other communication between NFs that is not directly related to the end-to-end data communication between edge applications. This category includes session management, routing or authentication. For instance, PPP session management, BGP routing and RADIUS authentication in a BRAS NF are examples of control plane workloads. When compared to data plane workloads, control plane workloads are expected to be much less intensive in terms of transactions per second, while the complexity of the transactions might be higher. This generally means lower load in R/W operations from/to memory and in I/O operations, and, at the same time, potentially higher load in CPU computing resources per packet (although the global load in CPU resources is expected to be lower since the packet rate is also smaller).
- **Signal processing workloads**, which cover all NF tasks related to digital processing such as the FFT decoding and encoding in a C-RAN Base Band Unit (BBU). These tasks are expected to be very intensive in CPU processing capacity and high delay-sensitive.
- **Storage workloads**, which cover all tasks related to disk storage, from the non-intensive logging of a router, to more intensive R/W operations such as the ones in a network probe (high load in write-only disk operations) or in a CDN cache node (high load in R/W operations from/to disk, high load in I/O operations related to access to content in SANs or NAS).

Figure 4 shows a summary of the different workloads.

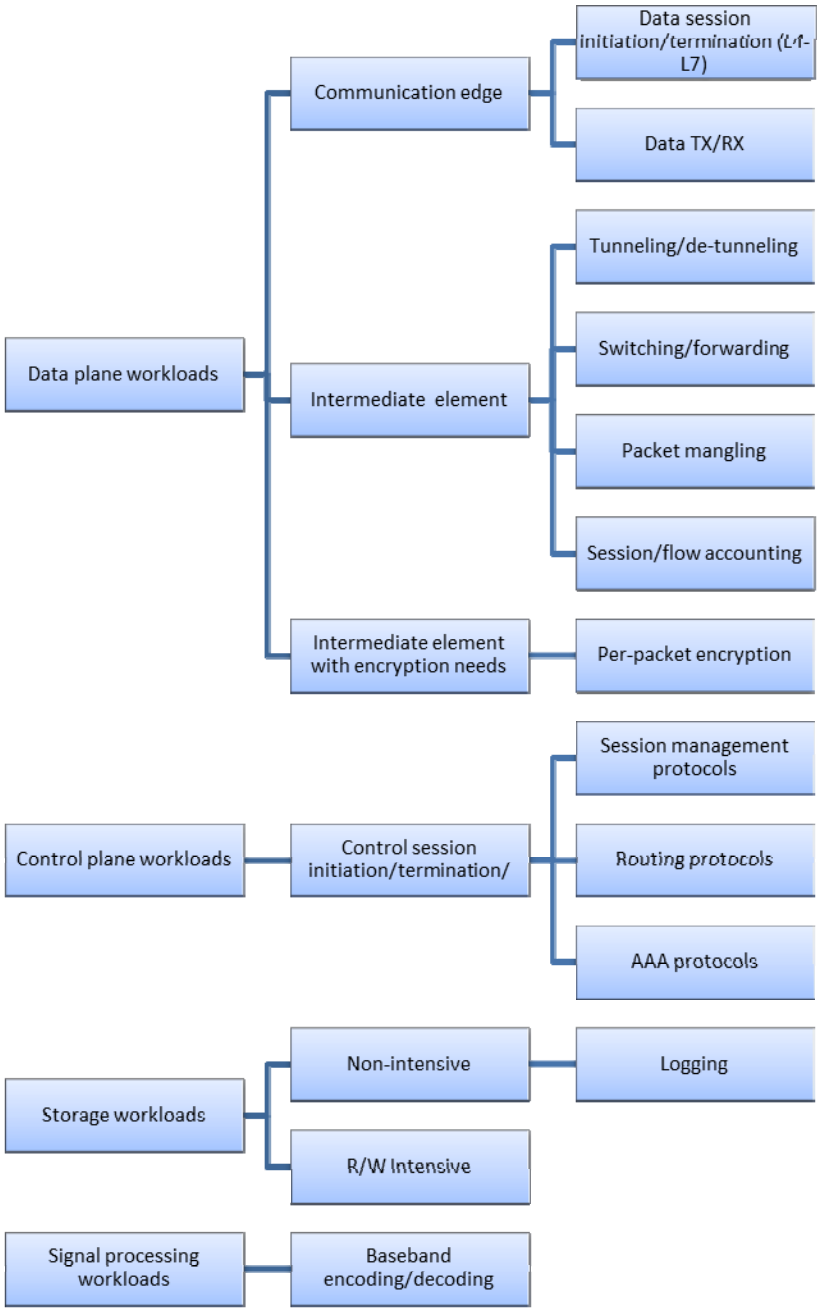


Figure 4: Workload classification

Most of these workloads happen to be present in several VNFs. Table 1 gathers, for each NFV use case in [2], the different comprised workloads.

Table 1: Workloads and relation to NFV use cases

		Data plane			Control plane			Signal processing	Storage	
Use case	VNF	Edge NF	Intermediate NF	Intermediate NF with encryption	Routing	Authentication	Session management	Signal processing	Non-intensive	R/W Intensive
1. NFVI as a Service	N/A									
2. Virtual Network Platform as a Service	N/A									
3. VNF as a Service	E-CPE (Enterprise-CPE)		X		X	X			X	
	PE		X	X	X	X			X	
	FW		X						X	
	DPI		X							X
4. Virtualisation of Mobile Core Network and IMS	MME				X		X		X	
	S-GW			X	X				X	
	P-GW		X		X	X			X	
	PCRF				X		X		X	
	SGSN			X	X		X		X	
	GGSN		X		X	X	X		X	
	P-CSCF		X		X	X	X		X	
	S-CSCF				X	X	X		X	
	I-CSCF				X		X		X	
	MGCF				X		X		X	
	AS	X					X		X	
5. Virtualisation of Mobile Base Station	BBU						X	X		
6. Virtualisation of the Home Environment	RGW		X		X	X			X	
	STB	X							X	
7. Service chains	N/A									
8. Virtualisation of CDNs	CDN Cache Node	X								X
	CDN Controller				X		X		X	
9. Fixed Access Network Functions Virtualisation	OLT		X		X			X	X	
	DSLAM		X		X	X		X	X	
	ONU		X					X	X	
	ONT		X		X			X	X	

As discussed, running tests focused on a specific workload of a given VNF simplifies the task, thus allowing to extend the conclusions to other VNFs where similar workloads are present.

For each of these tests, the methodology is shown in figure 5. As depicted in the figure, the methodology is based on an iterative experimentation process through tests, and the comparison of the test results in both bare metal and virtualisation environments.

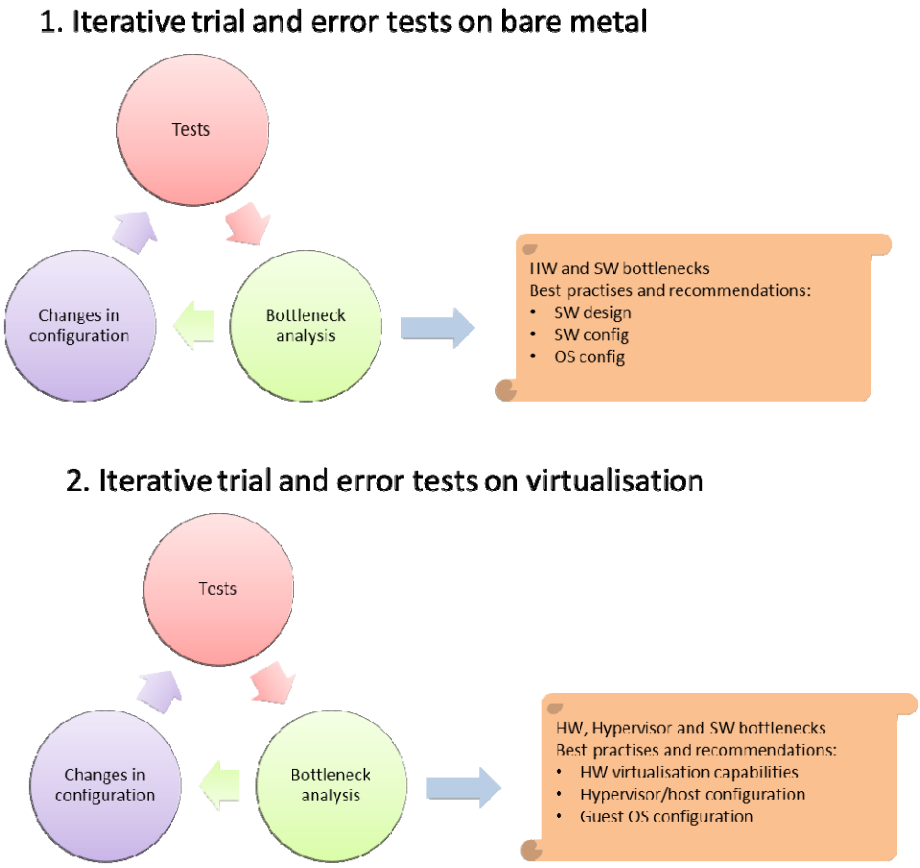


Figure 5: Methodology for bottleneck identification

First, tests are run on bare metal (that is, without virtualisation) in order to identify the relevant bottlenecks in the HW, as well as the best practises for SW design and SW configuration. Tests follow a common iterative trial-and-error methodology: after each trial, a bottleneck analysis is done; from that bottleneck analysis, a set of changes is proposed (SW design, SW config, OS config); then, these changes are applied and a new trial is performed. With each trial and error, the aim is to improve the performance metrics (e.g. packets processing throughput) as much as possible. As a result of this iterative process, the actual HW bottlenecks emerge, and it is possible to provide a set of best practises and recommendations related to HW itself, SW design, and SW configuration.

Once this analysis has been completed, a similar methodology is followed in the virtualised environment. In this case, the bottleneck analysis should take into account the virtualisation capabilities of the processor (memory, I/O, etc.), the virtualisation capabilities of the I/O device (e.g. SR-IOV), the hypervisor type and its configuration and the guest OS configuration. Again, with each trial-and-error cycle, the aim is to improve the performance metrics as much as possible taking into account that the target goal is the performance on bare metal. As a result of this second iterative process, it is expected to determine new bottlenecks in HW and in Hypervisor, and to provide a set of recommendations related to HW virtualisation capabilities, hypervisor/host configuration, and guest OS configuration.

In summary, the methodology can be seen as a two-step process:

- **Step 1**. Reach the maximum possible performance in bare metal, which should allow determining the actual HW and SW bottlenecks.
- **Step 2**. Reduce as much as possible the gap between the virtualised environment and the bare metal environment. This should allow identifying new bottlenecks related to virtualisation capabilities. In this regard, it should be noted that the gap might change from a virtualisation environment to another one (e.g. from a full-virtualisation environment to a para-virtualisation environment) and even from an hypervisor to another, although the qualitative effects from the interaction with HW are expected to be equivalent.

Annex C gathers test methodologies applicable for the performance evaluation of control plane and data plane workloads. In addition, one will notice that NFV introduces additional variables such as distributed deployment across multiple physical servers and failure detection via multiple NFV orchestration components such as EMS, VNF Manager and VIM. The ability for DUT components to reside as multiple VMs further makes VM migration scenarios possible as part of operational maintenance or resource optimization. This creates a need for testing these NFV specific scenarios to measure the impact of these variabilities on QoE and SLAs for various services and applications offered by the DUT. Some of these methodologies to test NFV-specific scenarios are also listed in annex C.

6 Bottleneck analysis and relevant features for high and predictable performance

This clause presents the main conclusions for the tested workloads, highlighting relevant HW and SW features and how they affect performance. The present document covers specifically the analysis of data plane workloads in intermediate elements such as a BRAS. A detailed description of the tests results can be found in annex C.

6.1 Data plane workload in an intermediate element

This clause discusses relevant HW and SW features to build and deploy VNF/VM Images dealing with data plane workloads such as packet switching/forwarding and header encapsulation/de-encapsulation. These workloads are present in VNFs such as a BRAS, a P-GW, or a PE node.

First, the relevant features to run VNFs dealing with this kind of data plane workloads on bare metal (e.g. including non-virtualised OS based software) are presented in clause 6.1.1. These relevant features have been identified from experimental tests of a simplified IP edge node (i.e. BRAS) developed for the purpose. In these tests, it was possible to achieve throughput figures in the order of magnitude of PNFs in the market. As discussed in annex C, the key is to remove the bottlenecks in the I/O and memory access to achieve that the VNF bottleneck is located in the CPU. Along clause 6.1.1, the different HW and OS features necessary to preserve this behaviour once the VNF is deployed are enumerated.

Clause 6.1.2 discusses the relevant features to obtain the highest possible performance on a fully-virtualised environment. These features were identified from experimental tests on the same BRAS prototype running on a VM. Tests followed the iterative trial-and-error procedure taking as target the throughput figures obtained in the bare metal tests. Same techniques as bare metal were applied on the virtualised environment to remove the common bottlenecks in the I/O and memory. However, a new bottleneck arose due to the need of double memory address translations in the virtualisation environment. After applying optimizations such as the HW support for Virtualisation, the gap with the bare metal was only due to the lack of HW virtualisation support for memory translation of large I/O pages. This limitation was solved in the latest processor generations gap, so that the performance in the virtualised environment is negligible when compared to bare metal.

6.1.1 High and predictable performance on bare metal

There are a number of features from processor, memory, I/O devices, the OS, and the SW code itself which make the difference to achieve a high and predictable performance in bare metal.

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

SW design is also of utmost relevance to take advantage of all capabilities offered by multi-core processor architectures. Aspects such **as pipelining, multi-threading and deadlock avoidance** should be part of any well-designed VNF, and these become essential in VNFs dealing with data plane workloads. Moreover, design should be oriented to exploit the HW architecture appropriately. In this regard, the **deterministic allocation of threads in cores/CPUs** can improve dramatically performance for some workloads, as threads which communicate each other and exchange intensively information might benefit from sharing the same cache memory. Moreover, **memory can be deterministically allocated** as close as possible to the core/CPU where the thread is running, thus avoiding inefficient memory trips. Some of these SW features such as the specific allocation of threads in CPUs and memory allocation might be needed for those VNFs needing high performance in I/O and memory lookups, while they could be optional for most of the control plane VNFs.

The **Operating System** is the glue between the HW and the SW; a first abstraction layer which hides much of the HW complexity to the SW. Thus, it provides features such as a multi-process runtime execution environment, memory management, independent memory access from different processes with overlapping memory addresses, file system management, etc. Advanced HW features can be appropriately exploited by SW as long as the OS supports them. This is the case, for instance, of large memory pages. Large memory pages reduce the time needed to translate from process' logical memory pages to host's physical memory pages by reducing the number of steps for that translation. Even more important, these translations can be cached in an internal processor cache, the TLB (Translation Lookaside Buffer), obtaining faster translation speeds when they are already cached. Besides, the translations for large pages benefit from an improved cache hit ratio since the same translation is shared by a higher number of logical memory addresses. It should be noted that large memory pages (and their TLBs) can be used in two types of memory access operations: from/to the I/O devices and from/to the processor. In fact, **support for large memory pages and TLB caches with large pages support** are present in several commercial processors for both types of memory access operations, and OSs such as Linux and Windows can exploit them as long as they are properly configured.

In some situations, however, the OS may not be able to exploit these techniques and, on the contrary, may introduce an additional overhead that limits the performance. For instance, the **OS task scheduling** is usually helpful and allows a fair CPU usage by the processes; however, for high performance applications, it may lead to frequent context changes, which penalize cache hit ratio in memory-intensive workloads (e.g. VNFs with high I/O performance requirements). As mentioned before, having control of which threads run on which cores and keeping that behaviour stable over time may improve significantly performance. In this regard, the OS provide ways to **isolate CPUs** from the task scheduling algorithm.

In other situations, some OS features might be explicitly bypassed by the SW, either using well known techniques in SW development of real-time applications, or leveraging on specialized libraries:

- Use of independent memory structures per thread, avoiding the need of using OS locks.
- Inter-thread communications can be based on internal pipeline memory structures instead of OS-based communications.
- Whenever I/O performance needs to be maximized, kernel data plane stacks may be bypassed, requiring for the SW modules to provide this I/O logic themselves, and read and write directly from/to memory without the help of the OS.
- Specialized SW libraries can help allocating physical pages on closer NUMA nodes, thus avoiding inefficient memory trips.

In general, whenever I/O and memory access are relevant for the overall performance of the VM Instances, techniques relying on bypassing the OS, its I/O mechanisms and its interruptions may become essential. However, even in those cases, **the OS still plays a key role to create the initial runtime environment for the VNF SW**.

Finally, **the way that network cards and other peripherals interact with the memory and OS also has a strong impact on performance**. Processors that support **DMA (Direct Memory Access)** enhance performance in I/O operations by allowing peripheral devices to access system memory directly, with no explicit intervention of the processor. This allows the CPU to do other operations while the transfer is in progress, and, once finished, an interruption from the DMA controller arrives to the CPU. Nevertheless, in spite of the improvement with respect to the architectures where no DMA was available, the interruption management by CPU required in this scheme still leads to inefficiencies for I/O intensive workloads. These I/O interruptions can be avoided by means of **drivers that make use of polling**.

Additionally, other advanced HW features such as **direct I/O access to processor cache** (see annex B for more details), allowing to write and read directly to the processor cache, can be exploited by network cards, making intensive communications between memory and I/O devices much faster.

Table 2 summarizes the features described previously, distinguishing between HW capabilities, SW design and configuration, OS support and configuration, or support from the I/O devices.

Table 2: Features that improve performance on bare metal

	HW	OS support and configuration	SW design and configuration	I/O device support and configuration
Processor architecture	x			
Extended instruction set (e.g. cryptography)	x			
Clock rate	x			
Size of data caches	x			
Memory access speed	x			
Memory latency	x			
Inter-processor bus BW	x			
Number of cores	x			
Large pages support	x	x	x	
IO Large pages support	x	x	x	
TLB cache with large pages support	x			
IO TLB cache with large pages support	x			
Size of TLB caches	x			
Interrupt affinity	x			
Layered memory cache	x		x	
Deterministic allocation of threads in CPUs			x	
Deterministic memory allocation			x	
Independent memory structures per thread (no shared structures)			x	
Inter-thread communication through memory pipeline structures			x	
CPU isolation		x		
Polling-mode drivers			x	
DMA	x			x
Direct I/O access to processor cache	x			x
Flow affinity/steering by I/O devices				x
NIC acceleration capabilities (e.g. TCP Segmentation Offloading)				x

Rows highlighted in grey represent the features that specifically depend on the HW or the OS. Features such as layered memory cache or deterministic memory allocation also depend on HW support, but can be taken for granted in current general purpose processors.

6.1.2 High and predictable performance on a virtualised environment

When moving from a bare metal to a virtualised environment, it should be taken into account that the same features and capabilities that were exploited in the previous clause should also be available from the perspective of the virtual machine. Additionally, the virtualisation environment introduces a new element in this equation, the hypervisor, which might add extra distortions which should be avoided or, at least, minimized. It should be noted that the penalties introduced by the hypervisor in some cases (e.g. control plane workloads) could be less relevant (even negligible) than in others (e.g. data plane workloads).

A hypervisor is a SW layer that creates an intermediate abstraction hiding the physical HW resources (CPU, memory, disk space, IO devices…) and slicing them into virtual HW resources. In this way, it is possible to run several OSs (guests) on top of those virtual HW slices, which effectively may run over the same physical HW. Some of the original hypervisor functions are the following:

- Hiding the HW and presenting a uniform virtual HW to the guest OS.

- Managing interruptions, mapping HW interruptions from/to the appropriate guest OS.
- Translating virtual memory addresses to physical memory addresses. Direct access to host physical memory pages from the guest OS is not possible. Instead, the guest OS uses guest-physical memory pages and the hypervisor is in charge of the translation from those guest-physical memory pages to host-physical memory pages (and vice-versa). As part of this translation, I/O memory access should also be translated, which means that DMA operations might also end up relying on the hypervisor.
- Offering a restricted CPU instruction set to the guest OS, thus making the behaviour of a guest OS stable through different HW platforms (e.g. emulation).

Due to this role of intermediary, the hypervisor might introduce additional penalties in performance if the environment is not properly controlled: double memory page translation, interrupt remapping, translation of instruction sets, etc. In the past, those penalties could be relevant since all the burden fell on the hypervisor. As hypervisors were evolving, those penalties were reduced; for instance, instruction emulation could be avoided when the guest instruction set is a subset of the original instruction set. Moreover, in the last years, virtualisation support has been added to general purpose processors, thus being possible to avoid total or partially that the burden lies on the hypervisor and which, instead, is transparently managed by the processor HW. Some key features of this virtualisation technology are the following:

- New instructions to reduce the number of VM exits under certain operations.
- Second-level address translation services (see HW support for Virtualisation in annex B for more details), so that CPU can interact directly with the virtual memory, bypassing the hypervisor. These translation services can also support translation of large pages, so that large pages can be used in virtual machines.
- IOMMU or translation services for I/O (see HW support for Virtualisation in annex B for more details). This includes:
    - Second-level memory address translation services for I/O, so that I/O devices can read/write directly from/to memory, backed by an IOTLB processor cache, thus bypassing the hypervisor (PCI pass-through). It might include translation services for large I/O pages, so that large I/O pages can be used in virtual machines.
    - I/O interrupt remapping
- Extension of processor memory caches with new fields to avoid cache eviction with VM exits, thus allowing the same cache to be shared by several VMs without flushing it with every context change.
- Extension of processor TLB caches with new fields so that the same TLB cache can be shared by several VMs avoiding cache eviction with every context change.

Additionally, I/O devices can implement additional features that can improve performance in a virtualised scenario:

- I/O device multiplexing (PCI-SIG SR-IOV) allows a single PCIe device to be split into several PCI devices (termed 'virtual functions' - PCIe-VF - in PCI-SIG) with different PCI configuration spaces, interrupts, I/O memory spaces, etc. Each of these PCIe-VFs can be assigned directly to a VM, which can interact directly with the device and vice versa, thus largely bypassing the hypervisor and avoiding the need of SW-based contention mechanisms.
- Sorting and classification of packets in independent TX/RX queues for virtual machines allows a single VM to manage directly I/O interruptions, thus avoiding one interruption for each packet at the hypervisor virtual switch (which manages potential NIC contention among VMs) to sort and classify packets.

HW support for virtualisation in processors and NICs improves significantly performance on a virtualised environment. In general, these features are intended to compliment as much as possible the hypervisor, in order to either bypass it or reduce previous hypervisor penalties. This does not mean that we should eliminate the hypervisor; on the contrary, **the hypervisor is essential to create the initial runtime environment for every virtual machine and make VNF deployment much easier, given the abstraction over the HW that it provides**.

At the same time, in a virtualised environment, it is mandatory that SW can keep exploiting the same optimizations that were common in bare metal execution in order to preserve equivalent performance. In particular:

- The **deterministic allocation of threads in CPUs** should still be possible so that threads that communicate each other and exchange information can share the same memory cache.

- The **deterministic allocation of memory** should also be possible so that memory is located as close as possible to the CPU where the thread is running, thus avoiding far memory trips.

For that to happen, it is needed that the hypervisor allows the deterministic mapping/pinning/affinity between virtual HW resources (vCPU, vMemory) and physical HW resources (CPU, memory). The same applies to large memory pages, which should be supported in the hypervisor in order to be used in the guest.

Table 3 adds to the features described previously in clause 6.1.1 the new ones from clause 6.1.2, showing to which category of resources are related. With respect to table 2, those features highlighted in grey and bold represent either new features or previous features which also are related to the hypervisor.

Table 3: Features that improve performance on a virtualised environment

	HW	Hypervisor support	Guest OS support and configuration	SW design and configuration	I/O device support and configuration
Processor architecture	x				
Extended instruction set (e.g. cryptography)	x				
Clock rate	x				
Size of data caches	x				
Memory access speed	x				
Memory latency	x				
Inter-processor bus BW	x				
Number of cores	x				
Large pages support	**x**	**x**	**x**	**x**	
IO Large pages support	**x**	**x**	**x**	**x**	
TLB cache with large pages support	x				
IO TLB cache with large pages support	x				
Size of TLB caches	x				
Interrupt affinity	**x**	**x**			
Layered memory cache	x			x	
Deterministic allocation of threads in CPUs (pinning between vCPU and CPU)		**x**		**x**	
Deterministic memory allocation (pinning between vMemory and Memory)		**x**		**x**	
Independent memory structures per thread (no shared structures)				x	
Inter-thread communication through memory pipeline structures				x	
CPU isolation		**x**	**x**		
Polling-mode drivers				x	
DMA	x				x
Direct I/O access to processor cache	x				x
Flow affinity/steering by I/O devices					x
NIC acceleration capabilities (e.g. TCP Segmentation Offloading)					x
Instructions to reduce the number of VM exits under certain common operations	**x**	**x**			
Second-level address translation services	**x**	**x**			
Second-level address translation services for large pages	**x**	**x**			
Second-level address translation services for I/O	**x**	**x**			
Second-level address translation services for large I/O pages	**x**	**x**			
I/O interrupt remapping	**x**	**x**			
Extension of processor caches with new fields to avoid cache eviction with VM exits	**x**	**x**			
Extension of processor TLB caches with new fields to avoid TLB flushes	**x**	**x**			
Independent TX/RX queues for virtual machines	**x**	**x**			**x**
SR-IOV	**x**	**x**			**x**

7 Best practises and recommendations

This clause provides a list of recommendations for a sample Compute Host aimed at a "telco datacentre". It is recognized that the current recommendations are required for VNFs supporting data plane workloads and that a small portion of the recommended list might not be required in all cases of VNFs, such as VNFs related to control plane workloads.

7.1 HW architecture and capabilities

- **Processor HW requirements**
  - Large pages support
  - I/O large pages support
  - TLB cache with large pages support
  - IOTLB cache with large pages support
  - HW support for virtualisation (see annex B for more details). It includes the following capabilities:
    - Instructions to reduce the number of VM exits under certain common operations
    - Second-level address translation services
    - Second-level address translation services for large pages
    - Extension of processor caches with new fields to avoid cache eviction with VM exits
    - Extension of processor TLB caches with new fields to avoid TLB flushes
  - HW support for virtualisation in I/O operations - IOMMU - (see annex B for more details). It includes the following capabilities:
    - Second-level address translation services for I/O
    - Second-level address translation services for large I/O pages (NOT PRESENT IN ALL PROCESSORS)
    - I/O interrupt remapping
  - Direct I/O access to cache (NOT PRESENT IN ALL PROCESSORS)

It should be noted that most of these features are supported by current processors. The last feature which was not present before but is supported in current processors at the time where the present document is released, is the second-level address translation services for large I/O pages.

Besides these features, we understand that Simultaneous Multi-Threading (SMT) is recommended in order to achieve the highest performance as possible from the underlying HW. This does not mean that SMT should be used, but, at least, it should be available since it has proved its benefits in terms of performance in some use cases. It is up to the Network Operators and the VNF providers to use it or not.

7.2 Hypervisor capabilities

- **Hypervisor/host OS requirements**
  - Capability to exploit HW support for Virtualisation:

    For each of the previous processor features related to HW support for virtualisation, it is also required that the hypervisor can exploit them and has them enabled.
    - Second-level address translation services

  - Second-level address translation services for large pages
  - Second-level address translation services for I/O
  - Second-level address translation services for large I/O pages (NOT PRESENT IN ALL HYPERVISORS)
  - I/O interrupt remapping

  Newest versions of hypervisors support all these requirements. Besides being supported, the hypervisor should be able to expose if they have been enabled in the system BIOS.

- Additional capabilities:
  - No instruction emulation when the guest instruction set is a subset of the original processor instruction set
  - Deterministic allocation of threads in CPUs (or logical CPUs in case that SMT is enabled)
  - Deterministic memory allocation in NUMA nodes (CURRENTLY UNSUPPORTED)
  - Deterministic allocation of large pages in NUMA nodes (CURRENTLY UNSUPPORTED)

  The deterministic memory allocation in NUMA nodes for the different VMs is necessary in order to guarantee that the memory is located as close as possible to the CPU where the thread is running, thus avoiding far memory trips.

Again, it should be noted that all the features are supported by at least one known hypervisor at the time where the present document is released, although it might be not supported by all current hypervisors.

8 Templates for portability

This clause details the minimal list of features that the Compute Host Descriptor and VM Descriptor templates should contain to assure high and predictable performance.

8.1 Compute Host Descriptor (CHD)

This clause covers the list of capabilities that should be exposed by a Compute Host. It is possible that, from a later analysis, some of the capabilities can be assumed as common in all general purpose processors. In that case, those capabilities could be omitted.

8.1.1 List of capabilities

8.1.1.1 Processor capabilities

A key decision to selecting a Compute Host is to know if the processor model is suitable for VM Images dealing with data-plane workloads. In this regard, the **processor vendor and model** should be exposed, so that the following advanced information can be inferred:

- Large pages support
- I/O large pages support
- TLB cache with large pages support
- IOTLB cache with large pages support
- HW support for virtualisation:
  - Instructions to reduce the number of VM exits under certain common operations

  - Second-level address translation services
  - Second-level address translation services for large pages
  - Extension of processor caches with new fields to avoid cache eviction with VM exits
  - Extension of processor TLB caches with new fields to avoid TLB flushes
- HW support for virtualisation in I/O operations:
  - Second-level address translation services for I/O
  - Second-level address translation services for large I/O pages
  - I/O interrupt remapping
- Direct I/O access to cache
- Simultaneous Multi-Threading (SMT) or similar technologies
- Extended instruction set (e.g. cryptography)

In addition to this qualitative information, the vendor and model of the processor provides quantitative information on relevant performance features, such as the clock frequency or the size of the processor cache. As mentioned in the previous clauses, relevant features are:

- Clock frequency
- Processor cache size
- Core cache size
- TLB cache sizes
- IOTLB cache sizes

Both the quantitative and qualitative information can be derived from the processor vendor and model, and could, therefore, be obtained from a database. Optionally, all this information could be requested directly to the hypervisor/host by the MANO Functions.

In some cases, it is also necessary to know **whether those features have been enabled in the BIOS system**. This information can be learnt by the MANO Functions, either from a database that is updated every time a Compute Host is rebooted, at regular runtime intervals, or with each new VM Instance.

It should be noted that, based on the previous information, the MANO Functions will take a decision on whether or not a VM Image can be deployed on a given Compute Host. Depending on the internal logic of the MANO Functions, decisions could be based on simple comparisons (e.g. by establishing a ranking of processors based on benchmarking and comparing the minimum processor required by a VM with the one in the specific Compute Host), or on more complex algorithms that take decisions based on all relevant performance features (clock frequency, processor architecture, cache size, core size, TLB cache size, I/O TLB cache sizes, etc.)

8.1.1.2 Memory capabilities

The **RAM memory capabilities** (DDR2 vs. DDR3, number of channels, etc.) can be relevant in those applications that are memory intensive. Although it is possible to derive a set of supported memory speeds from the processor vendor and model, the fact is that the memory speed depends specifically on installed memory modules.

Again, this information can be learnt by the MANO Functions, either from a database that is updated every time a Compute Host is rebooted, or because the MANO Functions request it to the hypervisor/host with every VM Instance. Depending on the internal logic of the MANO Functions, this capability can be exploited to take decisions, or simply unused.

8.1.1.3 Hypervisor capabilities

As explained earlier in clause 7.2, a key decision to selecting a Compute Host is to know if the hypervisor is suitable for VM dealing with data-plane workloads. In this regard, the **hypervisor type and version** should be exposed, so that the following advanced information can be inferred:

- Capability to exploit HW support for Virtualisation:
  - Second-level address translation services
  - Second-level address translation services for large pages
- Capability to exploit HW support for virtualisation in I/O operations:
  - Second-level address translation services for I/O
  - Second-level address translation services for large I/O pages
  - I/O interrupt remapping
- Additional capabilities:
  - No instruction emulation when the guest instruction set is a subset of the original processor instruction set
  - Deterministic allocation of threads in CPUs (CPU pinning/affinity)
  - Deterministic memory allocation in NUMA nodes (memory pinning/affinity)
  - Deterministic allocation of large pages in NUMA nodes (large memory pages pinning/affinity)

Besides, in some cases, it might be also necessary to expose **whether those features have been enabled or not**.

In addition, the hypervisor should expose its **vSwitch capabilities** (e.g. whether or not it is a SW-accelerated vSwitch). In this way, the MANO Functions will be able to infer if a vSwitch interface can meet the VM requirements on interface BW.

8.1.1.4 Resource topology and availability

Typically in cloud scenarios, there are no requirements on the resource topology. However, in NFV scenarios, as mentioned before, there could be situations where CPU, memory and I/O devices should be requested not only in terms of number, but also in terms of proximity. For this reason, we need the Compute Host to expose the full topology of resources and their availability. For instance, in Linux, topology info can be discovered with tools such as Portable Hardware Locality [i.4].

Figure 6 shows an example of a resource topology together with the resource availability. The figure illustrates a situation where SMT is available. It should be noted that if SMT is not supported, a similar topology could be built without lack of generality.

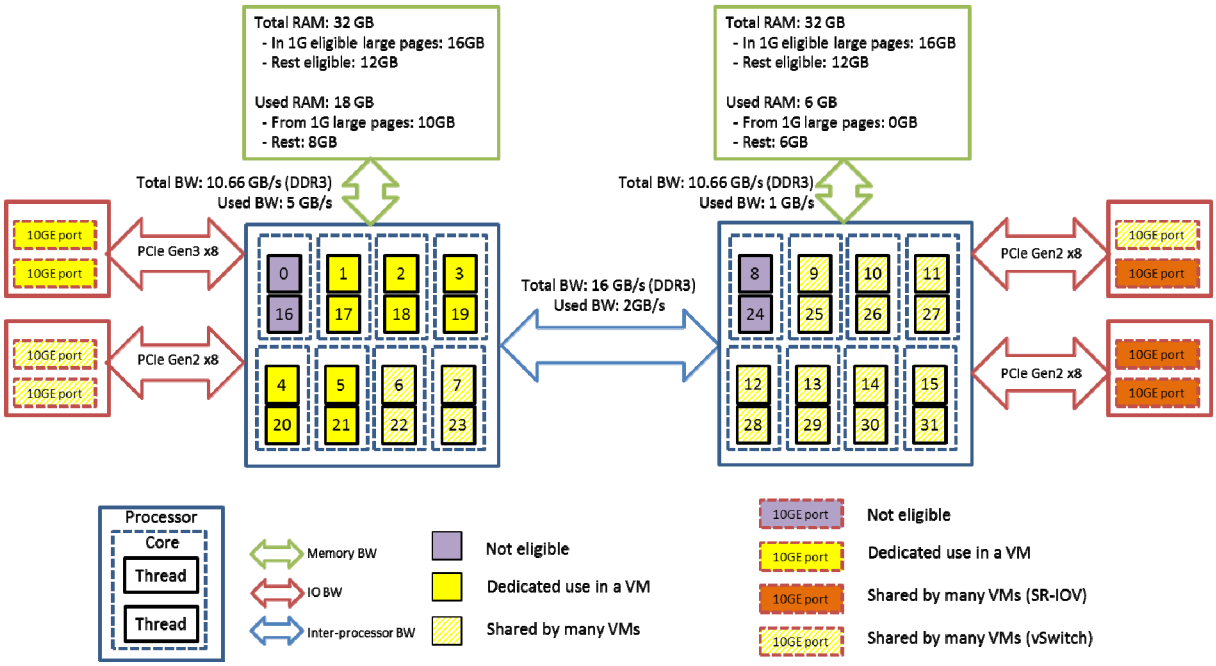


Figure 6: Example of resource topology and availability

The **resource topology and availability** can be specified in the following way:

- Number of NUMA nodes. A NUMA node is understood as a processor/socket and its associated memory and attached peripheral devices.
- For each NUMA node:
  - CPU:
    - CPU identifiers that belong to that NUMA node
    - Number of pairs of threaded CPUs
    - For each pair of threaded CPUs:
      - CPU identifiers that belong to that pair
    - For each CPU identifier:
      - Whether it is eligible by the VMs.
        - In case it is eligible, whether it is in use.
          - If it is in use, whether it is dedicated to one VM, or can be shared by many VMs.
            - In case of shared CPU:
              - Configured oversubscription ratio
              - Usage ratio
  - Memory:
    - Total memory of that NUMA node
    - Memory in large pages:
      - Size of large pages
      - Total number of configured large pages
      - Number of used large pages

        - Memory not in large pages:
            - Total memory
            - Whether it can be oversubscribed by many VMs or should be strictly assigned
            - Usage ratio (or free memory)
        - Memory BW (optional):
            - Total BW
            - Used BW
    - PCIe devices attached to that NUMA node. Specifically, we will focus on Network Interface Cards, which, for the sake of simplicity, will be assumed to be PCIe devices:
        - Number of NICs
        - For each NIC
            - Vendor and model. From the vendor and model, it would be possible to obtain the following information:
                - Type of PCIe device (PCIe2, PCIe3, etc.)
                - Number of lanes (x4, x8, etc.)
                - Number of ports
            - For each port:
                - PCI Device ID
                - Type/size (10G, 1G, etc.)
                - Whether it is eligible by the VMs
                    - If so, whether it is in use or not
                        - If it is being used, whether it is dedicated or shared by the VMs
                        - If it is shared:
                            - Shared with SR-IOV
                                - Number of configured Virtual Functions (VF)
                                - For each VF
                                - PCI Device ID
                                - Whether it is in use or not
                                - BW/Pacing for that VF (if configured)
                            - Shared through a vSwitch
                                - Whether the vSwitch is configured in bridge mode or NAT mode
                                - Number of virtual interfaces in use
- Bandwidth between NUMA nodes (optional):
    - Total BW
    - Used BW

The idea is to describe the resource topology in such a way that an appropriate assignment of resources can be made by the MANO Functions in order to guarantee that the VM will achieve a predictable performance. This resource topology should be exposed by the hypervisor and/or hypervisor manager based on its knowledge of the resources consumed by the VM.

The measurements on used BW for the memory channels and the inter-processor link are intended to provide a measurement of the potential impact over those VM Instances that make an intensive use of those buses. They are considered optional because there could be workarounds to control the impact among VM Instances, as will be detailed in the next clause.

8.2 Virtual Machine Descriptor (VM Descriptor)

8.2.1 Context on VM Image requirements

We are assuming scenarios where VNFs might be provided by VNF SW vendors as SW packages including a descriptor for the whole VNF, a set of VM Images and their corresponding VM Descriptors.

It is assumed that VNFs should be described as today's physical Network Functions, based on a set of "promised" functionalities and "promised" "Telco KPIs" (e.g. maximum input traffic, number of supported 10G interfaces, maximum number of sessions/s, maximum number of flows, etc.), specific for that VNF.

Internally, a VNF consists of several components or VMs. It is clear that the VNF Telco KPIs are directly related to the requirements that each VM Image demands from the Compute Host where they will be deployed. The way these VM Image requirements are obtained is not covered by the present document, although we foresee three different options, as depicted in figure 7.

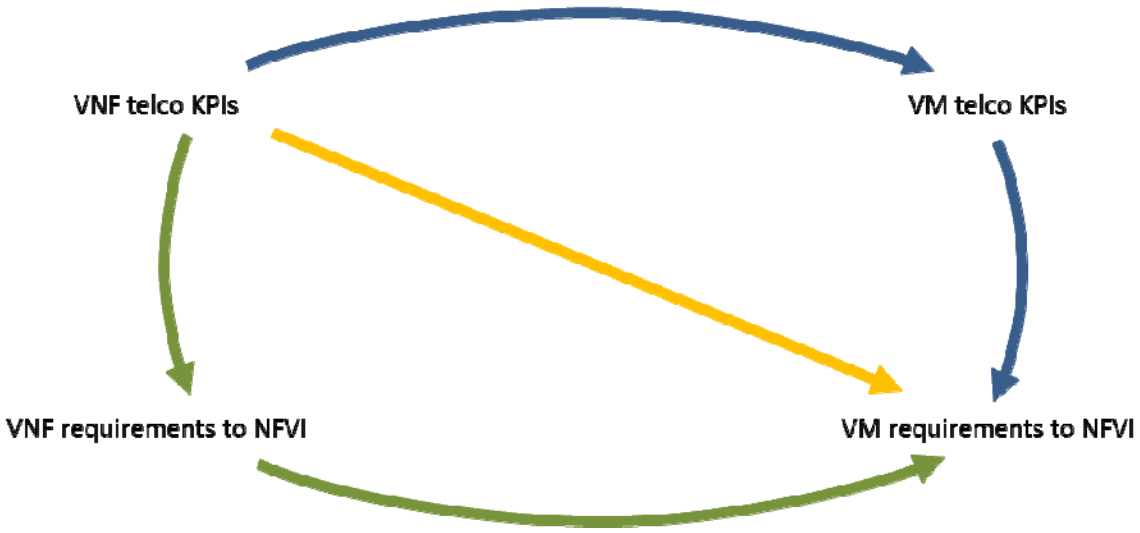


Figure 7: Relation between VNF telco KPIs and VM requirements

Typically, for most VNFs consisting only of a few VM Images, with clear role differentiation, VNF telco KPIs could be translated in similar KPIs for each of its VM Images (blue arrows). Depending on the specific set of VM Images telco KPIs, the VM Image requirements on number of CPUs, memory, etc. could be different. For instance, the memory size needed in table lookups will depend on the number of sessions/flows to be managed.

In any of the three options above, it is clear that VM Image requirements are directly related to the specific performance objectives (telco KPIs) of a VNF.

8.2.2 List of requirements

This clause describes the list of features that could be requested by a VM Image. These features range from the typical ones that are currently demanded for cloud VMs to the specific ones described in clause 8.2.2.4 for high performance scenarios. It is possible that, from a later analysis, some of the features can be assumed as common in all general purpose processors. In that case, those features could be omitted.

8.2.2.1 Processor requirements

Processor vendor and model. The VM Descriptor will declare the processor vendor and model for which the VM Image can achieve the expected performance. **From the model and vendor, it should be possible for MANO Functions to infer if other processors from the same vendor are suitable or not for that VM Image.** As mentioned before, that inference could be based on an established ranking of processors based on benchmarking or on more complex algorithms that take decisions based on all relevant performance features (clock frequency, processor architecture, cache size, core size, TLB cache size, I/O TLB cache sizes, etc.).

8.2.2.2 Memory requirements

The **RAM memory speed, together with the number of needed channels** can be relevant in those applications that are memory intensive. For that reason, it is recommended that those parameters are included.

8.2.2.3 Hypervisor requirements

- **Type and version of hypervisor**. The VM Descriptor will declare the hypervisor type (KVM, Xen, VMware, etc.) and version for which the VM can achieve the expected performance. It should be possible to know if the hypervisor supports the following capabilities:
  - Second-level address translation services
  - Second-level address translation services for large pages
  - Second-level address translation services for I/O
  - Second-level address translation services for large I/O pages
  - I/O interrupt remapping
  - No instruction emulation when the guest instruction set is a subset of the original processor instruction set
  - Deterministic allocation of threads in CPUs
  - Deterministic memory allocation in NUMA nodes
  - Deterministic allocation of large pages in NUMA nodes

  From the type and version, it should be possible for MANO Functions to infer if other hypervisors from the same vendor are suitable or not for that VM Image. Again, that inference could be based on an established ranking of hypervisors or on more complex algorithms.

- **Capabilities from the HW support**. Those capabilities that are related to HW support for Virtualisation should have been enabled by the hypervisor. For that reason, the VM Descriptor should declare which of those capabilities should be enabled. These can be joined for simplicity in 2 main capability sets:
  - **Capabilities to exploit HW support for Virtualisation:**
    - Second-level address translation services
    - Second-level address translation services for large pages
  - **Capability to exploit HW support for virtualisation in I/O operations:**
    - Second-level address translation services for I/O
    - Second-level address translation services for large I/O pages
    - I/O interrupt remapping

- **Virtual switch capabilities**. In case the VM Image required an interface from a SW-accelerated vSwitch, it should be provided by the hypervisor.

8.2.2.4 Number of resources and topology

- **Number of resources**
  - Processor:
    - Number of vCPUs
    - CPU oversubscription allowed
    - Restricted/unrestricted to a topology
  - Memory:
    - Memory size
    - Memory oversubscription allowed
    - Restricted/unrestricted to a topology
  - Interfaces:
    - Number of interfaces
    - For each interface:
      - Type/size
      - Dedicated/shared
      - Restricted/unrestricted to a topology
  - Storage:
    - Number of disks
      - Size
      - Type
      - Dedicated/shared
  - Other I/O devices:
    - PCI crypto devices

- **Topology of resources**

  Typically in cloud scenarios, there are no requirements on the topology or deterministic mapping of virtual and physical resources. However, in NFV scenarios, there could be situations where CPU, memory and I/O devices should be requested not only in terms of number, but also in terms of proximity. Figure 8 shows both a common set of requirements for a cloud VM (left) and a possible set of requirements for an NFV in the data plane (right). As mentioned before, the present document does not cover complicated storage scenarios where the storage BW could be a requirement. The requirements on used BW for the memory channels and the inter-processor link are intended to provide a measurement of the potential impact over those VMs that make an intensive use of those buses. As will be commented below, they are considered optional because there could be workarounds to control the impact among VMs.

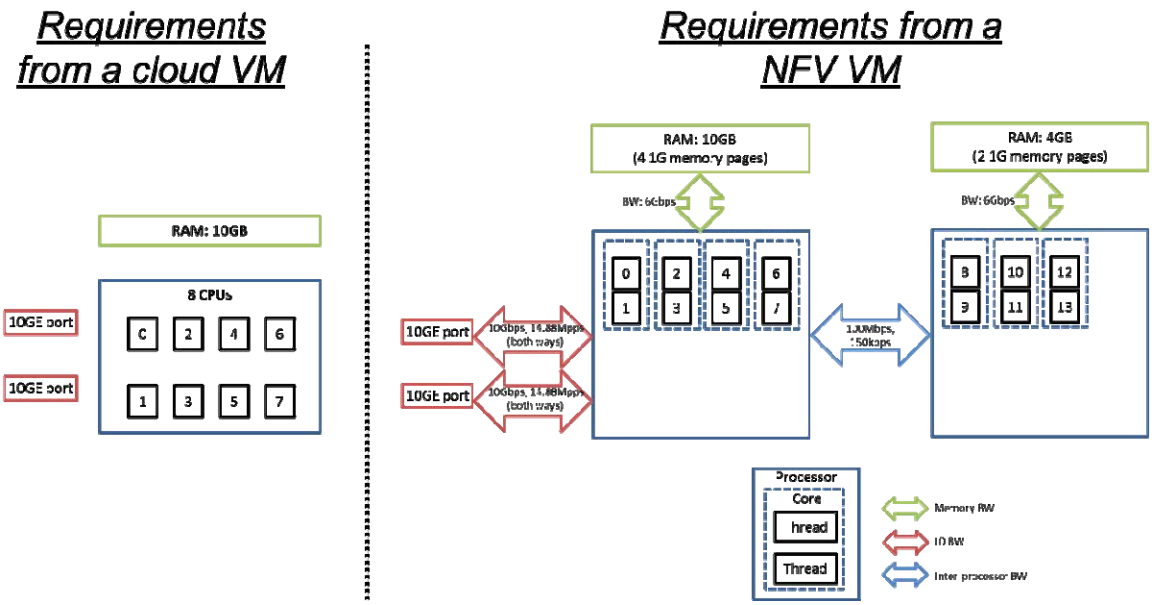


Figure 8: Comparison between a possible set of requirements for a cloud VM and for an NFV VM

The figure on the right illustrates a situation where SMT is requested. It should be noted that if SMT is not supported, a similar set of requirements could be used without lack of generality.

The topology of resources can be specified in the following way:

- Number of vCPU sets. A vCPU set is understood as a set of vCPUs that are located in the same processor, thus benefiting from sharing cache and avoiding far memory trips.
- For each vCPU set:
  - CPU requirements:
    - vCPU identifiers that belong to that vCPU set
    - Number of pairs of hyperthreads vCPUs
    - For each pair of hyperthreaded vCPUs:
      - vCPU identifiers that belong to that pair
  - Memory requirements:
    - Total memory required for the vCPU set
    - Size of large pages
    - Number of large pages
    - Required BW (optional)
  - Storage requirements (not covered in the present document)
  - Network Interfaces:
    - Number of interfaces
    - For each interface:
      - Required BW
      - Fully dedicated/Shared through SR-IOV/Shared through a vSwitch

- Bandwidth requirements for communication between vCPU sets (optional):
  - BW required for each pair of vCPU sets

As an example, below there is a possible implementation of the requirements from figure 8:

```
cpu-sets: 2
cpu-set 0:
       cpus:
              number: 8
              howtogroup: [0,1] [2,3], [4,5], [6, 7]
       memory:
              total: 10GB
              hugepages size: 1GB
              hugepages number: 4
       interfaces:
              number: 2
              interface 0:
                     speed: 10Gbps
                     bandwidth: 10Gbps
                     fully-dedicated
              interface 1:
                     speed: 10Gbps
                     bandwidth: 5 Gbps
                     shared-sriov
cpu-set 1:
       cpus:
              number: 6
              howtogroup: [8,9], [10,11], [12,13]
       memory:
              total: 4GB
              hugepages size: 1GB
              hugepages number: 2
       interfaces:
              number: 0
cpu-set-bw:
       cpu-sets: 1,2
              bandwidth: 100Mbps
```

8.2.2.5 Impact from/to other VMs

Typically in cloud scenarios, the impact of one VM over another VM running in the same host is controlled through global VM deployment policies. In telco datacentre scenarios, since the topology of resources might be relevant, another kind of policy might be necessary in order to restrict the number of VMs running on the same host. In this regard, the impact on processor data caches and TLB caches can become extremely relevant when I/O and memory access performance are key for the global behaviour of the VM Instance.

It becomes necessary a method to control the type and number of VMs that can run simultaneously on a host. Although a thorough study is still needed, we devise some lines of work:

- Option 1. To characterize the impact from/to other VMs through a benchmarking with a standardized set of VMs (memory intensive, CPU intensive, etc.). The result of that benchmarking would be explicitly included in the requirements on impact.
- Option 2. To measure the number of identical VMs that could run on the same platform with low performance degradation and use that number as a measurement of the impact.
- Option 3. To estimate usage of common resources (memory BW, inter-processor BW, data cache, TLB cache, etc.) under pessimistic conditions and use those figures as a measurement of the impact.

In addition, there could be a potential impact from the vSwitch utilization:

- The vSwitch introduces an additional burden in the host in terms of resource consumption (CPU, memory). In fact, the resources consumed by the vSwitch might be higher as the number of VMs grows.
- If several VMs use a vSwitch, there could be an impact on the interface bandwidth due to congestion in the vSwitch.

This ends the list of requirements for a VM, which, as seen, consists of processor and memory HW requirements, hypervisor requirements, number of resources, topology of resources and impact from/to other VMs.

It should be noted that the list of requirements in terms of number of resources, topology and impact can change from one processor to another: for instance, the number of necessary cores can be lower in a 2,7 GHz processor than in a 1,8 GHz processor. The same happens with hypervisors: one hypervisor might exploit HW support for large pages, while another hypervisor might not be prepared. Therefore, the VM Descriptor template might consist of different lists of requirements for a set of processors and hypervisors, from which it could be easy to extrapolate to other potential configurations e.g. via worst-case approaches (e.g. the same configuration with a processor with 75 % a reference frequency specified in the VM Descriptor would grant, as a minimum, 75 % the reference performance) or more sophisticated techniques.

Annex A (informative): Gap analysis of hypervisor and cloud OS

From the list of capabilities offered by a Compute Host, an analysis has been made to detect exposure gaps in hypervisors and cloud OSs. Even if the HW supports a capability, we need the hypervisor to expose that capability in order to use it, and the same happens at orchestration level with common management frameworks such as OpenStack, CloudStack or others.

Table A.1 shows the result of this gap analysis.

Table A.1: Gap analysis of hypervisor and cloud OS exposure

Performance & Portability Template				
Feature	**Required be exposed at Hypervisor**	**Required to be Exposed at Orchestrator**	**Description/Justification**	**Priority**
Hypervisor Enabled (e.g. KVM)	Yes	Yes	Hypervisor must be present and enabled before VM can be deployed to platform.	Required
IOMMU	Yes	Depends on cloud OS	An input/output memory management unit (IOMMU) enables vms to directly use peripheral devices, such as Ethernet, accelerated graphics cards, and hard-drive controllers, through DMA and interrupt remapping.	Required
Direct cache access for Ethernet adapters	Needs to be exposed	Needs to be exposed	Eliminate slow I/O memory access model by allowing Ethernet adapters to directly address host cache.	Required
Memory Available	Yes	Yes	Memory available on host platform. The amount of free memory on the platform has an impact on the number of VMs that can be deployed without using the same memory which would impact performance. There must also be enough free memory remaining for the host OS to function correctly.	Required
Large Pages Available	Currently not exposed	Yes	Number of large pages available. Large page pool will be taken from the Total Memory available on the system. Large pages which are free should be shown so as not to share the same large pages between VMs.	Required
PCI I/O device present (NIC)	Exposed using Device ID	Needs to be exposed	Type of NIC present will define features available (e.g. flow affinity, port pacing, mirroring etc).	Required
PCI Crypto device present	Exposed using Device ID	Needs to be exposed	Type of Crypto device and features available if required for application.	Optional
PCI Device attachment socket	Needs to be exposed	Needs to be exposed	Required to understand which CPU socket a PCI device is attached to. This is to ensure that PCI devices used with a VM are attached to the same socket that VCPUS are populated on. This also avoids the QPI bottleneck.	Required
PCI bandwidth limit	Needs to be exposed	Needs to be exposed	Required to be aware of Bandwidth limitations for PCI device slots	Required
SR-IOV device present	Yes	Needs to be exposed	Required to be aware of Virtual Functions available on the host platform	Required
CPU Model & Flags	Yes	Depends on cloud OS	Required to be aware of the CPU capabalities and hence portability i.e. pse36 for hugepages, vmx for hardware virtualisation extensions etc. CPU flags give no indication of cache layer size. Total cache size can be gathered from cpuinfo	Required
Software Accelerated OVS	Needs to be exposed	Needs to be exposed	Required if OVS avails of software acceleration for the dataplane e.g. Intel DPDK used with OVS	Optional
Core Availability	Needs to be exposed	Needs to be exposed	Need to be aware of which CPUs are available and not being used by a VM or Host OS. If Hyperthreading is present, should be aware of which are physical cores and logical cores. Both physical and logical core should be assigned to same VM VCPUs. Benefit avoids QPI interface by ensuring all VCPUs populated from same physical socket.	Required
Core Isolation	Needs to be exposed	Needs to be exposed	Need the ability to isolate cores to ensure they are not used by Host OS	Required
Core allocation (Socket)	Needs to be exposed	Needs to be exposed	Need to be aware of which socket available CPUs are attached to	Required
Kernel version	Needs to be exposed	Needs to be exposed	Kernel version has direct impact SW features (DPDK, OVS etc).	Required

Annex B (informative): Relevant technologies

This annex contains a list of relevant reference technologies mentioned along the document.

B.1 HW support for Virtualisation

HW support for Virtualisation is a set of features in COTS processors (e.g. Intel's® VT-x or AMD's® AMD-V, see note 1), intended to assist the Hypervisor so that some of the Hypervisor burden (mainly memory address translations and context switches) can be managed by the processor itself, thus improving performance in the VM. Some of the features are the following:

NOTE 1: Intel's® VT-x and AMD's® AMD-V are examples of suitable products available commercially. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of these products.

- New instructions to reduce the number of VM exits under certain operations.
- Second-level address translation services (e.g. Intel's *Extended Page Tables* or AMD's® *Rapid Virtualisation Indexing*, see note 2), so that CPU can interact directly with the virtual memory, bypassing the hypervisor. These translation services can also support translation of large pages, so that large pages can be used in virtual machines.

NOTE 2: Intel's Extended Page Tables (EPT) and AMD's® Rapid Virtualisation Indexing (RVI) are examples of suitable products available commercially. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of these products.

- Extension of processor caches with new fields to avoid cache eviction with VM exits, thus allowing the same cache to be shared by several VMs.
- Extension of processor TLB caches with new fields so that the same TLB cache can be shared by several VMs.

In addition, COTS processors include a set of features in the I/O Memory Management Unit (IOMMU) (e.g. Intel's® VT-d or AMD's® AMD-Vi, see note 3) to assist the Hypervisor in I/O operations, mainly in translation services. Some of the features are the following:

NOTE 3: Intel's® VT-d and AMD's® AMD-Vi are examples of suitable products available commercially. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of these products.

- Second-level memory address translation services for I/O, so that I/O devices can read/write directly from/to memory, thus bypassing the hypervisor. It might include translation services for large I/O pages, so that large I/O pages can be used in virtual machines.
- I/O interrupt remapping, which allows the interrupt to arrive directly to the VM bypassing the Hypervisor.

B.2 Direct I/O access to processor cache

Direct I/O access to processor cache (e.g. Intel's® Direct Data I/O, see note) is a feature in COTS processors which allows an I/O device to write and read directly to/from the processor cache. This can be exploited by network cards, making intensive communications between memory and I/O devices much faster.

NOTE: Intel's® Data Direct I/O (DDIO) technology 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.

B.3 Single Root I/O Virtualisation (SR-IOV)

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

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

B.4 Remote Direct Memory Access (RDMA)

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

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

B.5 Infiniband

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

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

Annex C (informative): NFV Test Methodologies

This annex describes the detailed test methodologies for testing network devices in a NFV environment. It also specifies the key focus areas for testing VNFs and where it is different from testing traditional networking devices.

C.1 Control Plane Testing

The control plane testing procedures for a network device depend on the functionality supported by the network device. For example, when validating an edge router, the functionalities to be tested would be the maximum number of routing sessions or MPLS tunnels supported, routing convergence time upon failure, session bring up time, route installation time and multi-protocol scale. Similarly for a BNG device, PPPoX, DHCP or L2TPsession capacity and session bring up rate need to be tested. Additional test scenarios with optional features such as authentication, DPI enabled or disabled can be defined to assess the performance impact of these features on session capacity and session performance.

C.1.1 Common Testing Methodologies for physical world and virtual world

The most significant metrics that are measured during the control plane validation are listed below:

- Scalability - Refers to the maximum number of control plane sessions that can be established. Examples include the number of PPPoX sessions or number of routing peers. For routers, number of routes per session and total number of routes in the routing table are also a measure of scalability.
- Session performance - Session ramp up and tear down rate.
- Message processing time - How fast the control plane messages are processed and long term min, max and average statistics.
- Fail-over convergence testing - Time for the network device to converge the routing path from primary to the secondary path, when a node or link failure is detected on the primary path.
- Functional Testing - May consist of session establishment, tear-down, traffic forwarding support, topology creation, installing control plane state, message exchange.
- Interoperability Testing - Verify inter-operation of the control plane protocol/protocols under consideration for implementations from different vendors.
- Conformance Testing - Verify if the implementation of control plane protocols is as per the specifications laid out by various standards organizations.

C.1.2 NFV specific control plane testing

Performance testing and results analysis for physical network devices are well-established and involve fewer variables because of the self-contained nature of the device under test.

In NFV environments, VNFs can be implemented across multiple physical servers, either via provisioning during service turn up or triggered at run-time by events or policies. This distributed implementation imposes new requirements on control plane testing that are described below.

C.1.2.1 Scalability

In NFV environments, operators can scale out (dynamically instantiate VNF components in new servers to increase performance) or scale in (delete VNF components from servers to reduce performance) in response to the varying network function performance needs. The scaling procedures are either triggered automatically, when certain resource usage, session counts or performance thresholds are exceeded, or manually, by operator intervention. Multiple NFV components including the VNF itself, the VIM and the MANO Functions are involved in the orchestration of the scaling out/in procedures.

C.1.2.1.1 Test Setup

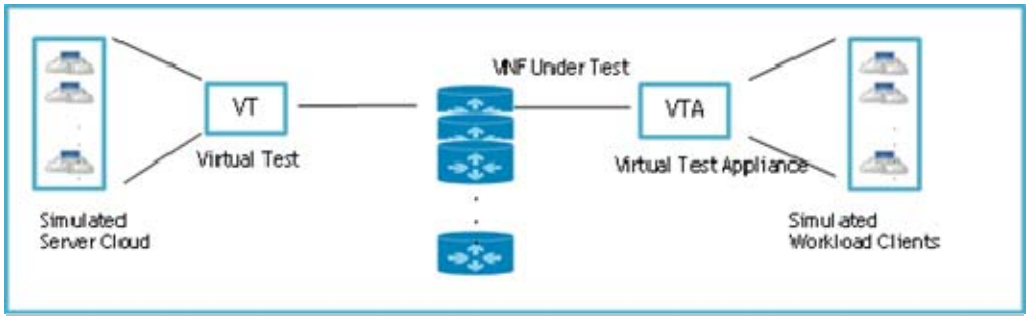


Figure C.1: VNF Auto-Scaling Test Topology

- Provision a VNF for DUT with default number of constituent VMs
- Provision two virtual test endpoints connected to the virtual DUT. In order to avoid the performance impact of resource sharing between DUT and virtual test boxes, install the virtual test endpoints on a separate physical server
- Configure test endpoints for L4-L7 stateful traffic with one endpoint as server and other as client
- Configure test endpoints for stateless traffic
- Configure one of the endpoints for testing the performance and scalability of control plane
- If the virtual DUT is a router, configure routing protocols such as OSPF, ISIS or BGP
- If virtual DUT is a BNG or an access device, configure access protocol such as DHCP or PPPoE

C.1.2.1.2 Test Procedure

Step 1 - Configure valid workload traffic on the Client and Server

Configure a traffic mix on the client and server including:

- HTTP
- FTP
- DNS
- Streaming video
- Telnet
- NTP
- SSH
- Syslog
- NFS

- Messaging
- VoIP
- Social Networks
- Unicast Video
- P2P
- CIFS
- Background Traffic

Step 2 - Baseline performance of the test bed without the DUT

- Run the base line test to benchmark the maximum number of transactions/second, average, minimum and maximum latency without the DUT in the test configuration

Step 3 - Run Baseline/Calibration Test and Record Results

- Disable the auto-scaling feature on DUT in order to base line the performance
- Run the baseline test to benchmark the maximum number of transactions/sec and average, min, max response time supported by DUT

Step4 - Test the auto-scale functionality of the VNF

- Enable auto-scaling feature on DUT
- Without stopping the traffic workload used in Step 2, increase the workload with 5 % additional simulated users
- Wait for VM provision time (default value - 10 minutes) for the traffic to ramp up and the VNF to provision additional VMs/resources to accommodate the additional traffic

Step 5 - Record the results

- Record the transactions/sec, average, min and max response time
- Record the total number of VMs instantiated by the VNF
- Record the NFVI resources used by the VNF (processor, memory, storage)

Step 6 - Benchmark the performance scale

- Repeat Steps 3 and 4 with 10 %, 15 %, 20 % simulated users

Step 7 - Benchmark control plane scale

- In this scenario, instead of L4-L7 traffic mix, configure the virtual test appliance to emulate the maximum number of control plane sessions
- Execute step 2 to baseline the control plane protocol scale with auto-scale disabled and then repeat steps 3-6 for the control plane functionality under consideration

In this example we are establishing the baseline by measuring the maximum number of simulated users or maximum number of sessions supported by virtual DUT (VNF) in the default configuration with auto-scaling disabled. The test can also be run by establishing the baseline against the SLAs desired for the service, when the corresponding policies have been configured on DUT. When the workload is exceeded beyond a threshold where the SLAs are no longer satisfied then auto-scale should be activated to procure additional resources to satisfy the SLAs and deliver the desired QoE.

C.1.2.1.3 Desired Results

- The successful transaction rate for the additional simulated users should be similar to the performance metrics for existing simulated users.
- The performance of the existing simulated users should not be affected when additional simulated users are added to the workload.

Benchmarking the NFVI resources and number of VMs required to auto-scale for various levels of additional load will provide a trend data and a measure of auto-scaling efficacy of the VNF.

C.1.2.2 Failover Convergence Testing

Convergence time is one of the key metrics for validating SLAs and high availability in service provider networks. The requirements for failover convergence times can be in the order of milliseconds depending on the services and applications that are under consideration. These constraints are equally valid in an NFV environment as well.

In NFV deployments, there is an added factor of variability where failover convergence time for a VNF can be impacted by the number of VNFs on the physical server that is converging to alternate routes. Convergence measurement involves the measurement of processing time of the trigger event in the control plane and the traffic switchover time. It is important in a multiple VNF deployment scenario that the convergence time of any VNF is not impacted by the other VNFs on the same physical server, so that the VNF continues to satisfy the SLAs for which it was provisioned.

C.1.2.2.1 Test Setup

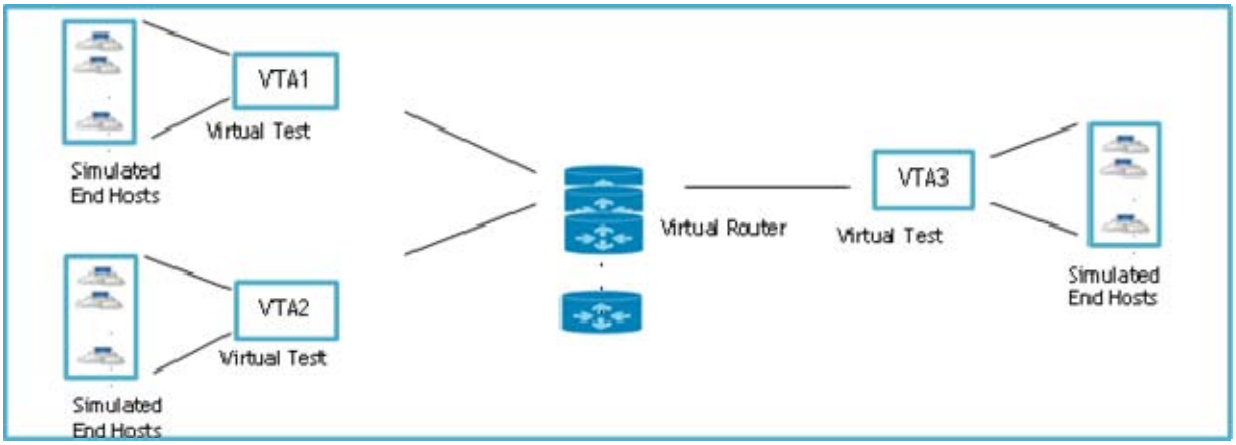


Figure C.2: Fail-over Convergence Test Topology

- Provision a VNF for DUT with routing protocol functionality.
- Provision two virtual test endpoints (VTA) connected to the virtual DUT. In order to avoid the performance impact of resource sharing between DUT and virtual test boxes, install the virtual test endpoints on a separate physical server.
- Configure the pre-determined routing protocol on the two virtual test appliances and advertise same set of routes from both virtual test appliances. Use a preferred metric in the routes advertised from one of the virtual test appliances (primary next hop/path).
- Configure L3 traffic between endpoints advertised by VTA 3 and endpoints advertised by VTA 1.
- Configure liveness detection mechanisms such as BFD protocol on the VNF as well as VTA1 and VTA2.

C.1.2.2.2 Test Procedure

Step 1 - Advertise Routes

- Start routing protocol on the two virtual test appliances and advertise the routes to the simulated end hosts. Same prefix addresses are advertised from both the virtual test appliances but the routes advertised by VTA 1 should be preferred metric.

Step 2 - Verify the routes

- Send L3 traffic from VTA 3 destined for simulated hosts advertised by VTA 1.

Step3 - Disable used to detect link or the liveness mechanism node failures

- If Layer 1 detection is used then disable the link between DUT and TA 1 and if heartbeat mechanism such as BFD is used then stop the transmission of heartbeat messages from TA 1.
- Also stop the transmission of routing protocol packets from TA 1.

Step 4 - Record the results

- Verify that the traffic for all the endpoints advertised by TA 1 switches to TA 2.
- Record the time (t1) when the event for failure was triggered in Step 3.
- Record the time (t2) when the first packet of first traffic stream arrives on secondary port (TA 2).
- Record the time (t4) when the first packet of the last traffic stream arrives on secondary port (TA 2).

Step 5 - Calculate the convergence result as follows

- Convergence Time = Timestamp of 1st packet of last stream on backup (t4) - Timestamp of 1st trigger event (t1).
- Traffic Switchover Time = Timestamp of 1st packet of last stream on backup (t4) - Timestamp of 1st packet of first stream on backup (t2).
- Trigger event duration = t3 - t1.

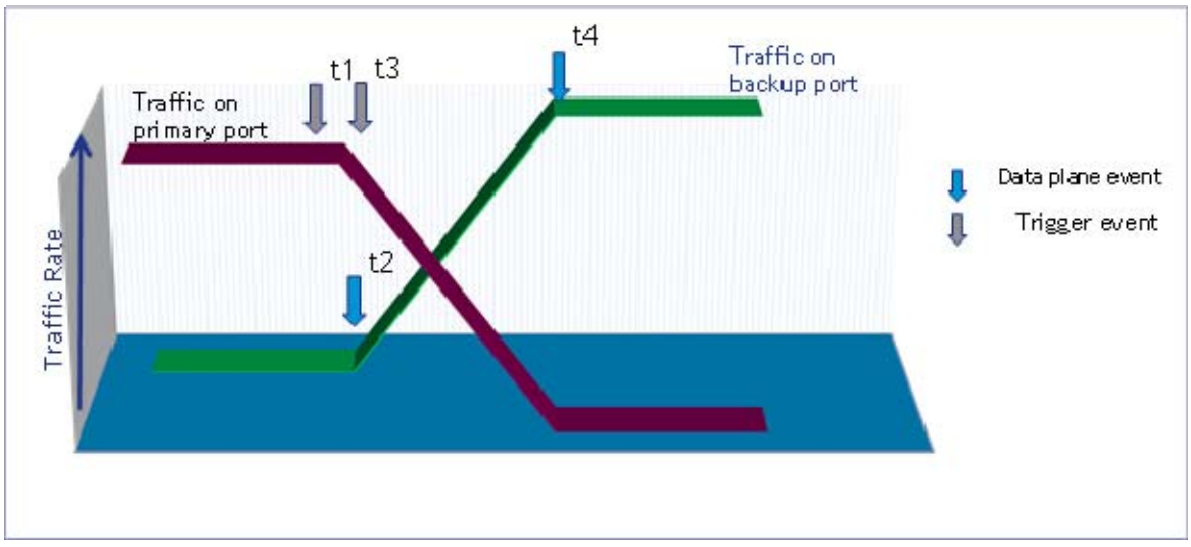


Figure C.3: Fail-over Convergence Measurements

Step 6 - Scale number of endpoints for convergence

- Increase the number of endpoints or routes advertised by TA1 and TA2 in increment of 10 % from the original value.
- Repeat steps 1-5 to measure the convergence for all the endpoints in the configuration.
- Repeat the test steps when increasing with increments of 10 % till the specified convergence SLAs for the VNF under test are exceeded. This measure will benchmark the maximum number of routes a VNF can support to meet the desired convergence time SLAs.

Step 7 - Benchmark convergence measurement when the number of VNFs is scaled higher

- Double the number of VNFs (virtual DUTs) on the same physical server.
- Repeat the test by increasing the number of VNFs and running the convergence test on all the existing VNFs at the same time.

This step will provide us with the effect of scaling the number of VNFs on convergence time. A graphical representation of the trend will provide us with useful information on the acceptable number of VNFs to satisfy the SLAs for convergence time.

For service provider networks there are multiple time critical applications where the SLA dictates a convergence time under 50 milliseconds. In such scenarios it becomes more important that the convergence measurement is accurate and the control plane processing time is taken in account when calculating the convergence time. Thus it is convenient that the test tool supports accurate time measurements for control plane as well as data plane and provides a mechanism to synchronize the reference clocks for the testers used in the test topology.

Step 8 - Bare metal measurement

- Repeat the test steps 1-5 for running the test on bare metal.

C.1.2.3 VM Migration

This concept is specific to virtual environments only. VM migration from one Compute Host to another may be required because of oversubscription of resources, operational maintenance and service continuity upon link or node failure. Due to distributed VNF implementation in NFV, VM migration may be further categorized in three different ways:

- Migration of a constituent VM from one Compute Host to another
- Migration of a VNF (and its constituent VMs) in service chain from one Compute Host to another
- Migration of VM or VNF across data centers

C.1.2.3.1 Test Setup

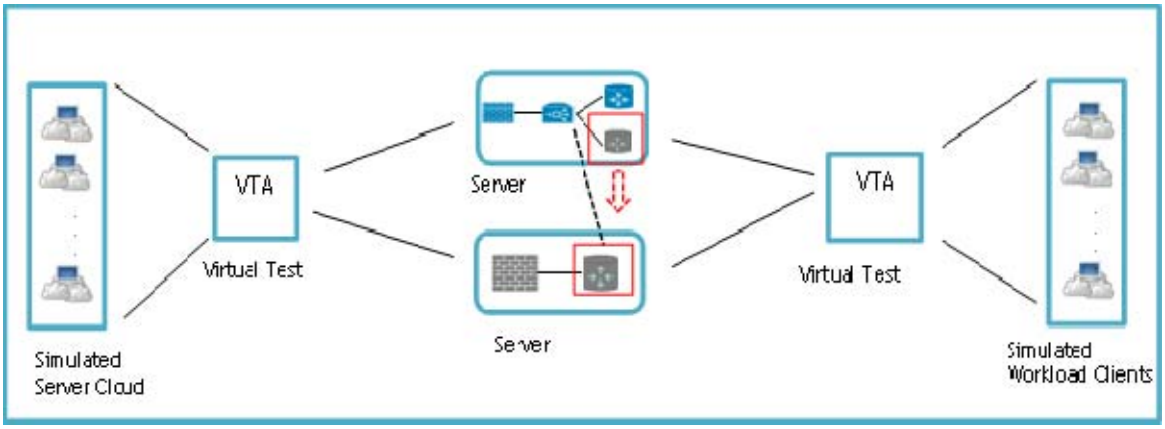


Figure C.4: VM Migration Test Topology

- Provision a service chain with the VNF under testing and other VNFs such as router, firewall and load balancer on Server 1 and provision Server 2 with any other VNFs.
- Provision two virtual test endpoints connected to the virtual DUT. In order to avoid the performance impact of resource sharing between DUT and virtual test boxes, install the virtual test endpoints on a separate physical server.
- Configure the VNFs so that data traffic for the simulated endpoints are forwarded from VTA 1 to VTA 2 in both directions.
- Configure L3 traffic between VTA 1 and VTA 2.

C.1.2.3.2 Test Procedure

Step 1 - Advertise Routes

- Start routing protocol on the two virtual test appliances and advertise the routes to the simulated end hosts.

Step 2 - Verify the routes

- Send L3 traffic from VTA 1 destined for simulated hosts advertised by VTA 2.
- Wait for the traffic to reach steady state.
- Run tests following RFC 2544 [i.5].

Step3 - Move VNF from Server 1 to Server 2

- Move the VNF as shown in the topology diagram from Server 1 to Server 2. Note that the connections with other VNFs in the service chain should be maintained.
- Run tests following RFC 2544 [i.5].

Step 4 - Record the results

- Record the any packet loss during the VM migration from Server 1 to Server 2.
- Tabulate the results when all the VNFs were provisioned in the same server against when one of the VNFs was in a different server.

Step 5 - Different VM migration scenarios

- Repeat Steps 3 and 4 for the following scenarios:
  - One of the constituent VMs of the router VNF is moved from Server 1 to Server 2 when both Server 1 and Server 2 are part of the same datacenter.
  - One of the constituent VMs of the router VNF is moved from Server 1 to Server 2 when Server 1 and Server 2 are part of different datacenters (scenario for Datacenter Interconnect).
  - VNF is moved from Server 1 to Server 2 when Server 1 and Server 2 are part of different datacenters (scenario for DataCenter Interconnect).

C.1.2.3.3 Result Analysis

The objective of this test is to measure the VM migration duration/Service Disruption time due to VM migration and to compare the performance or QoE for the end user in the scenario before and after the VM migration.

C.2 Data Plane Testing

Forwarding performance benchmarking for data plane has been well-defined and understood in the industry via benchmarking standards such as RFC 2544 [i.5], RFC 2889 [i.6], RFC 3918 [i.7] and RFC 3511 [i.8].

Most of the methodologies and specifications for data plane performance benchmarking are equally applicable to physical and virtual network devices.

C.2.1 NFV specific Data Plane Testing

C.2.1.1 Example of performance benchmarking of a DPI device in different configurations

Some physical dedicated network devices have a dedicated DPI engine but NFV opens up the possibilities of optimizing performance by using a shared DPI engine across multiple VNFs residing on the same physical server. The impact of a shared DPI engine in a NFV scenario is not well understood and warrants performance benchmarking for the two possible configurations (shown below) of DPI engine functionality.

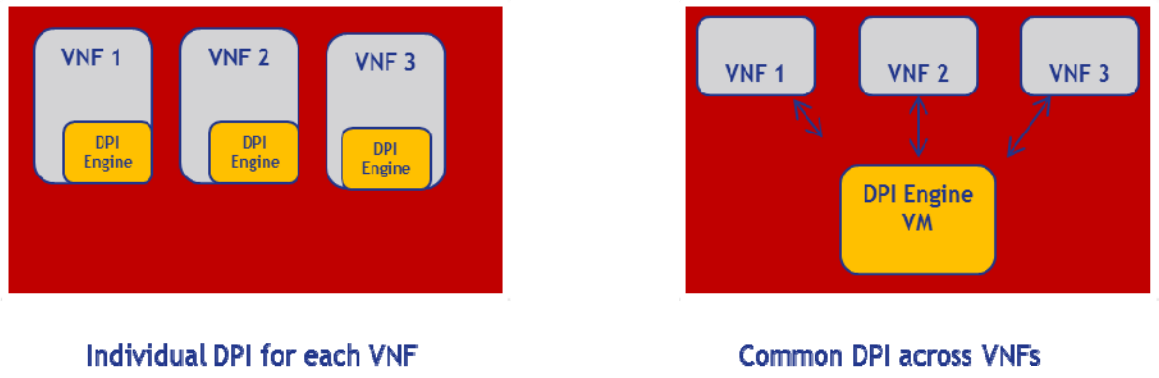


Figure C.5: DPI configurations

C.2.1.1.1 Test Setup

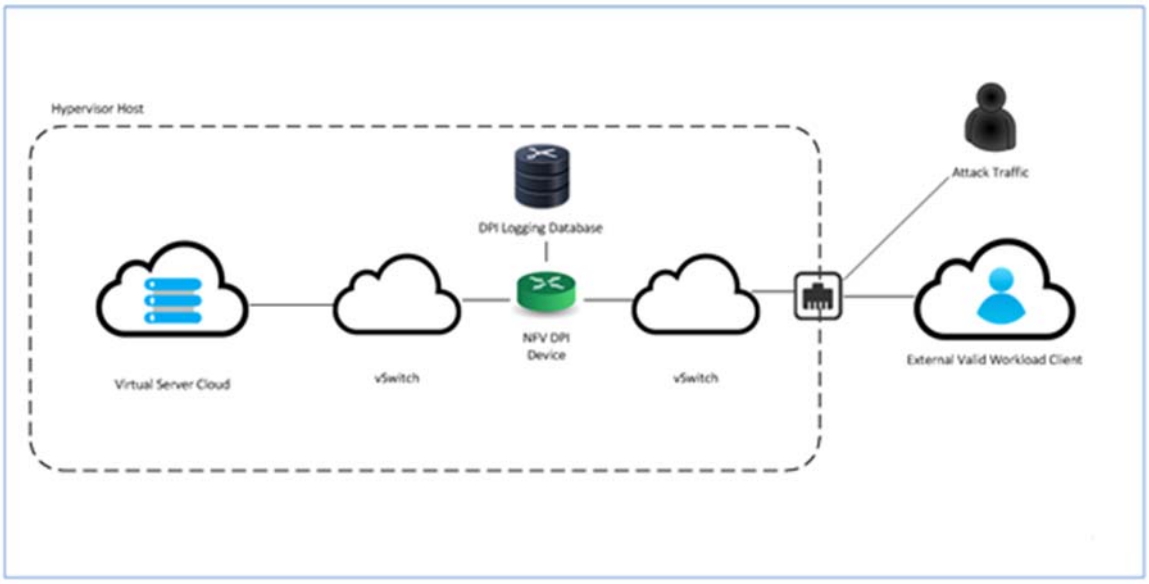


Figure C.6: DPI Test Topology

Here are the steps necessary before beginning the test case:

- Setup NFV DPI DUT:
  - This test case will use the DUT in three modes: bypass, transparent, and routed modes.
  - Provision the NFV DUT initially in bypass mode; specifically the DUT passes traffic through the DUT and no other functions.
- Instantiate two vSwitch networks, called client_vswitch and server_vswitch. Connect the DUT to the appropriate virtual NICs on the DUT. The client_vswitch network should also include a physical Ethernet port on the hypervisor.
- Configure a virtual test endpoint for server traffic termination, connected to the server_vswitch.
- Configure a physical appliance test port for client traffic; connect the physical test port to the provisioned hypervisor Ethernet port attached to client_vswitch.
- Attach the test GUI to the client and server ports.

C.2.1.1.2 Test Procedure

Here are the steps to perform this test case:

Step 1 - Configure valid workload traffic on the Client and Server

Configure a traffic mix on the client and server including:

- HTTP
- FTP
- DNS
- Streaming video
- Telnet
- NTP
- SSH
- Syslog
- NFS
- Messaging
- VoIP
- Social Networks
- Unicast Video
- P2P
- CIFS
- Background Traffic

The client population should be at least a total pool of 1 000 client IP addresses or more.

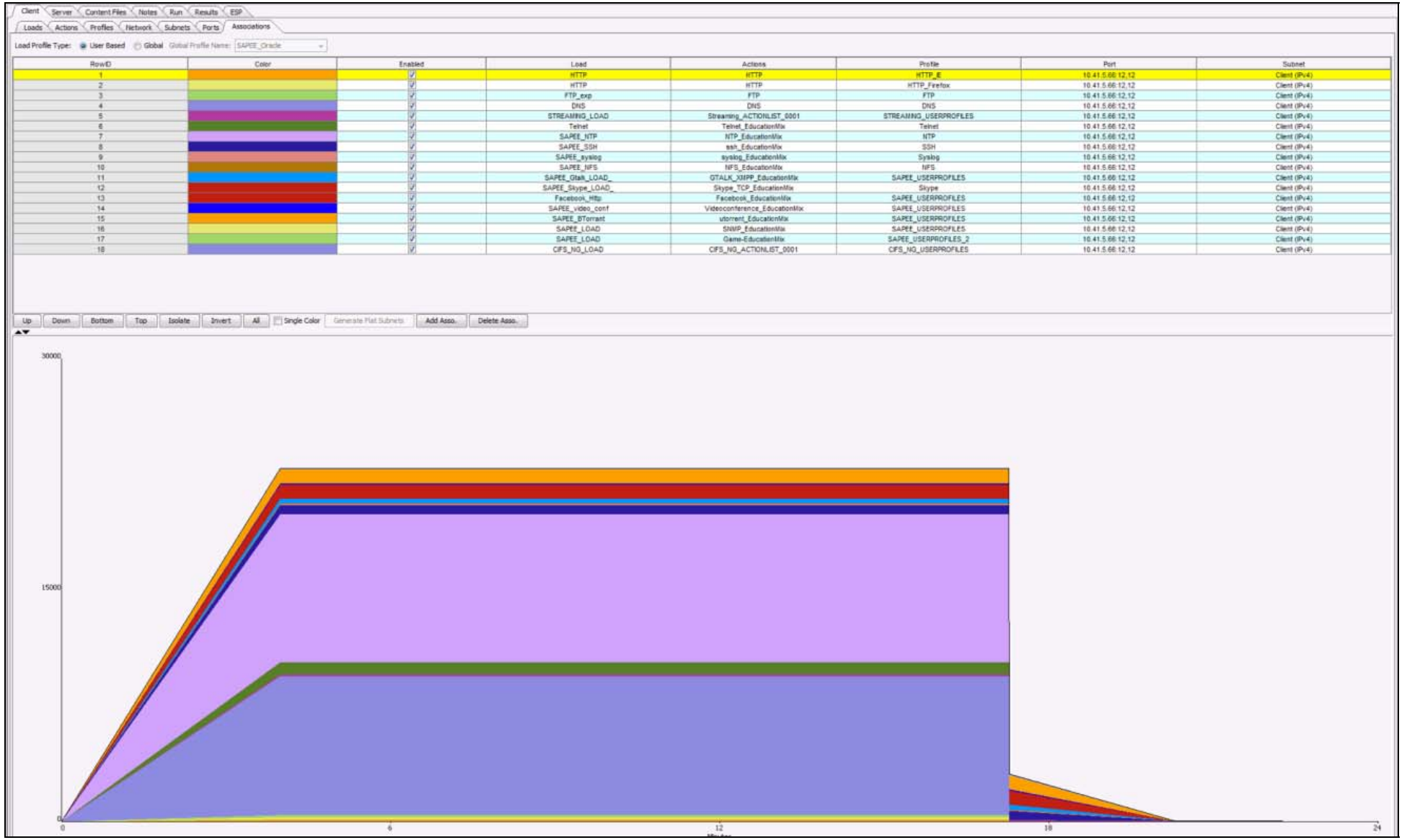


Figure C.7: Example of protocols and distributions

Step 2 - Setup Traffic Loading and total test duration

- Traffic should ramp up to approximately 20 k simulated users per second, with no more than 50 active users per protocol. The total duration of the test should be 20 minutes with approximately 4-minute ramp-up and ramp-down and a trail off period of 4 minutes.

Step 3 - Run Baseline/Calibration Test and Record Results

- The hypervisor and vSwitch infrastructure are first baselined. This allows the user to measure and compare results independent of the hypervisor. Be sure to place the DUT into bypass mode, run the test.

- If actual vs. expected traffic varies, or there are Layer 2-7 errors, then reduce the number of constrained active simulated users per protocols by -5 simulated users for each protocol until there is no loss of performance and no protocols errors (including TCP retransmissions, dropped packets, or timeouts).

- Run the test again and measure the achieved network bandwidth at 10 minutes.

- Record the results. This result is considered the be the 'Ideal case', and shale be used as a comparison metric for measuring impact of the DUT upon traffic.

Step 4 - Run Test with DUT configured for Inspection/Logging Off/No Attack

- Configure the DUT to inspect traffic, but not log events. Run the test and measure the bandwidth achieved at 10 minutes.

- Compare the generated transactions to the measured transactions in the DUT counters:

	Test	Count	Transactions (Sub-Commands included)								Response Time (ms)		
	Profile	URL	Average Successful Per Second	Attempted	Successful	Unsuccessful	Aborted	Percent Successful	Percent Unsuccessful	Percent Aborted	Minimum	Maximum	Average
Transaction Summary	HTTP_IE_18	8	70	102141	102016	125	0	99.87	0.12	0.0	0.0	3946.0	709.622
	HTTP_Firefox_19	8	73	105965	105850	115	0	99.89	0.1	0.0	0.0	3994.0	713.815
	FTP_20	2	18	26786	26579	207	0	99.22	0.77	0.0	139.0	298574.0	2618.495
	DNS_21	4	11433	16592104	16522001	70103	0	99.57	0.42	0.0	0.0	7404.0	463.344
	STREAMING_USERPROFILES_22	5	0	685625	0	685625	0	0.0	100.0	0.0	0.0	63259.0	1374.468
	Telnet_23	1	485	704802	701802	3000	0	99.57	0.42	0.0	1.0	100001.0	1023.498
	NTP_24	1	2515	3651875	3634190	17685	0	99.51	0.48	0.0	1.0	1259016.0	3291.06
	SSH_25	1	242	352794	351121	1673	0	99.52	0.47	0.0	3.0	1259471.0	1828.314
	Syslog_26	1	39	56742	56742	0	0	100.0	0.0	0.0	4.0	4.0	4.0
	NFS_27	1	23	33822	33749	73	0	99.78	0.21	0.0	1.0	100001.0	1352.588
	SAPEE_USERPROFILES_28	1	33	48405	48141	264	0	99.45	0.54	0.0	3152.0	1266713.0	7537.54
	Skype_29	1	370	540903	535807	5096	0	99.05	0.94	0.0	30993.0	131026.0	32890.242
	SAPEE_USERPROFILES_30	1	45	65971	65936	35	0	99.94	0.05	0.0	4.0	1158134.0	511.61
	SAPEE_USERPROFILES_31	1	11	17154	17072	82	0	99.52	0.47	0.0	4.0	1226133.0	5398.056
	SAPEE_USERPROFILES_32	1	27	39584	39551	33	0	99.91	0.08	0.0	26970.0	101498.0	27613.535
	SAPEE_USERPROFILES_33	1	7	10314	10277	37	0	99.64	0.35	0.0	64.0	100001.0	875.52
	SAPEE_USERPROFILES_2_34	1	7	10314	10314	0	0	100.0	0.0	0.0	20408.0	21777.0	20795.685
	CIFS_NG_USERPROFILES_35	1	2	4066	4065	0	1	99.97	0.0	0.02	17.0	335486.0	2899.854
	HTTP_IE_0	8	72	104212	104082	130	0	99.87	0.12	0.0	0.0	3990.0	713.95
	HTTP_Firefox_1	8	72	104656	104541	115	0	99.89	0.1	0.0	0.0	3893.0	713.343
	FTP_2	2	20	29792	29582	209	1	99.29	0.7	0.0	139.0	146478.0	2702.939
	DNS_3	4	11445	16608812	16539420	69392	0	99.58	0.41	0.0	0.0	7358.0	463.245
	STREAMING_USERPROFILES_4	5	0	685600	0	685600	0	0.0	100.0	0.0	0.0	75802.0	1376.212
	Telnet_5	1	489	709955	707010	2945	0	99.58	0.41	0.0	1.0	100001.0	1013.72
	NTP_6	1	2511	3647311	3629755	17556	0	99.51	0.48	0.0	1.0	1259017.0	3297.539
	SSH_7	1	241	350542	348871	1671	0	99.52	0.47	0.0	3.0	1259451.0	1846.336
	Syslog_8	1	39	56742	56742	0	0	100.0	0.0	0.0	4.0	4.0	4.0
	NFS_9	1	23	34143	34078	65	0	99.8	0.19	0.0	1.0	100001.0	1334.978
	SAPEE_USERPROFILES_10	1	32	47223	46953	270	0	99.42	0.57	0.0	3152.0	1256638.0	7768.558
	Skype_11	1	371	541256	536149	5107	0	99.05	0.94	0.0	30993.0	131014.0	32887.934
	SAPEE_USERPROFILES_12	1	45	66237	66197	40	0	99.93	0.06	0.0	4.0	1238189.0	509.653
	SAPEE_USERPROFILES_13	1	11	16742	16651	91	0	99.45	0.54	0.0	4.0	1265514.0	5647.956
	SAPEE_USERPROFILES_14	1	27	39598	39569	29	0	99.92	0.07	0.0	26970.0	110949.0	27608.68
	SAPEE_USERPROFILES_15	1	7	10314	10268	46	0	99.55	0.44	0.0	64.0	100001.0	956.988
	SAPEE_USERPROFILES_2_16	1	7	10314	10314	0	0	100.0	0.0	0.0	20408.0	21781.0	20793.766
	CIFS_NG_USERPROFILES_17	1	2	3128	3126	0	2	99.93	0.0	0.06	30.0	336801.0	4787.621
	Totals	80		46115944	44548521	1567419	4	96.6	3.39	0.0			

Figure C.8: Sample Test Results

If the transaction count is not the same, or you see any failed transactions (illustrated here), or the QoE of HTTP average response time is greater than 1,5 × the baseline, then reduce the active SimUser per protocol and rerun this step until these conditions are met.

- Log this result.

Step 5 - Run Test with DUT configured for Inspection/Logging On/No Attack

- Configure the DUT to inspect traffic and log events. Run the test and measure the bandwidth achieved at 10 minutes. Compare the generated Transactions to the measured transactions in the DUT counters.
- If the transaction count is not the same, or you see any failed transactions (Illustrated here), or the QoE of HTTP average response time is greater than 1,5 × the baseline, then reduce the active SimUser per protocol and rerun this step until these conditions are met.
- Log this result.

Step 6 - Configure DDoS and Key Vulnerability Attacks

- The user should now add to the configuration file a mix of 5 DDoS and vulnerability attacks and activate them in the configuration. In addition, the user should turn on tester side logging of attacks.

Step 7 - Run Test with DUT configured for Inspection/Logging Off/Attacks On

- Configure the DUT to inspect traffic, do not log events, and turn attacks on. Run the test and measure the bandwidth achieved at 10 minutes. Compare the generated transactions to the measured transactions in the DUT counters.
- If the transaction count is not the same, or you see any failed transactions (illustrated here), or the QoE of HTTP average response time is greater than 1,5 × the baseline, or the DUT Attack counters do not equal the tester attack counters, then reduce the active SimUser per protocol and rerun this step until these conditions are met.
- Log this result.

Step 8 - Run Test with DUT configured for Inspection/Logging On/Attacks On

- Configure the DUT to inspect traffic, log events, and turn attacks on. Run the test and measure the bandwidth achieved at 10 minutes. Compare the generated Transactions to the measured transactions in the DUT counters.
- If the transaction count is not the same, or you see any failed transactions (illustrated here), or the QoE of HTTP average response time is greater than 1,5 × the baseline, or the DUT attack counters do not equal the tester attack counters, then reduce the active SimUser per protocol and rerun this step until these conditions are met.
- Log this result.

Step 9 - Performance scale testing of multiple VNFs (Separate DPI engine for each VNF)

- Double the number of VNFs on the physical server that need DPI functionality.
- Repeat steps 1 through 8.
- Increase the number of VNFs again and Repeat Steps 1 through 8 till the maximum number of VNFs on the server reaches its maximum benchmark.

Step 10 - Performance scale testing of multiple VNFs (Common DPI engine for all VNFs)

- Double the number of VNFs on the physical server that need DPI functionality.
- Repeat steps 1 through 8.
- Increase the number of VNFs again and repeat steps 1 through 8 till the maximum number of VNFs on the server reaches its maximum benchmark.

C.2.1.1.3 Desired Result

The desired result for this test scenario is no difference between the baseline and any combination tested.

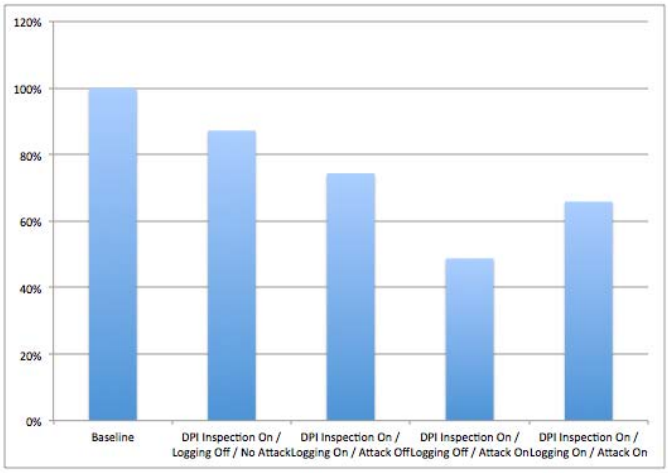
C.2.1.1.4 Measured Result

Table C.1: Test result metrics

Test Step/Post Step	Metric & Unit	Purpose
Baseline Bandwidth	Throughput in Mbps	Baseline of Hypervisor
DPI Inspection On/Logging Off/No Attack	Throughput in Mbps	Measures the cost of inspection of traffic and logging
DPI Inspection On/Logging On/No Attack	Throughput in Mbps	Measures the cost of inspection of traffic and logging
DPI Inspection On/Logging Off/Attack On	Throughput in Mbps	Measures the cost of inspection with workload and attacks enabled
DPI Inspection On/Logging On/Attack On	Throughput in Mbps	Measures the cost of inspection with workload, attack traffic, and inspection on
Trend performance data for scaling of VNFs with DPI functionality	Throughput in Mbps	Plans are afoot to make the source code publically available

C.2.1.1.5 Analysis

The user should normalize the bandwidth of each combination relative to the baseline by dividing the bandwidth result of the measured result to the baseline, and then represent each measured result as a percentage of the baseline. By definition, the baseline is 100 %. The difference between the measure result percent and the baseline is the cost impact of the NFV DPI device, under that condition. The results could be represented using a bar chart and table, including the baseline. The table should include the percent and the measured bandwidth. Below is a sample result.



	Baseline	DPI Inspection On / Logging Off / No Attack	DPI Inspection On / Logging On / Attack Off	DPI Inspection On / Logging Off / Attack On	DPI Inspection On / Logging On / Attack On
Percent	100%	87%	74%	49%	66%
Achived Mbps	234234	204234	174234	114234	154234

Figure C.9: Test results with DPI Inspection enabled

Scoring the result of the DUT, take the smallest percentage measured vs. the baseline, and use a grading table such as the following:

Table C.2: Result Score Criteria

Final grade	Condition (smallest percentage impact vs. baseline)
A	Score is 100 % ≥ Measured Result ≥ 95 %
B	Score is 94 % ≥ Measured Result ≥ 80 %
C	Score is 79 % ≥ Measured Result ≥ 70 %
D	Score is 69 % ≥ Measured Result ≥ 60 %
F	Score is < 60 %

C.3 Benchmarking hypervisors

It is important to understand that when measuring the performance of a virtual network device, we are measuring the cumulative performance of many entities involved: the performance of the VM, the hypervisor and the NFVI resources. Any comparison between performances of an entity from different vendors should be done only when other entities are kept constant. For example, if we want to compare the performance of VNFs provided by different ISV, then we should use the same hypervisor and NFVI resources when running the performance benchmark tests.

Following the same procedure, hypervisors from different vendors can be benchmarked using the existing benchmarking tests while using the same VNFs and NFVI resources. Some considerations are listed below:

- Traditional benchmarking for hypervisors at the lower granular level, which includes metrics such as CPU operations, RAM operations and disk operations, are equally applicable. One should be careful to use the right load as defined by vendor independent organizations such as SPEC-Virt to obtain accurate benchmark results. It may be useful to derive a model to estimate the number of VNFs of a particular type supported by the hypervisor based on the CPU, memory and disk I/O performance benchmarks of the VNF and the hypervisor.
- Maximum number of VMs supported on a hypervisor is not a sufficient benchmark. Instead we have to benchmark the maximum number of VMs supported on the hypervisor under traffic such that the QoE for the services supported by the VMs is within acceptable limits and the required SLAs are satisfied.
- When benchmarking forwarding performance of the hypervisor, direct I/O pass-through and SR-IOV could provide different results.
- Some hypervisors allow optimization of VM scalability and CPU utilization at the expense of higher latency by reducing the number of virtual network interrupts. Some of the applications and services have very low latency requirements and it re-emphasizes that the VNF scale benchmarking is done under load when service SLAs and QoE are satisfied.
- Some hypervisors have support for optimization features such as "Split Rx mode", "Large page support", "Virtual Network Interrupt Coalescing" and "Jumbo frame support". It is important that when we compare the benchmark results for various hypervisors, we understand the trade-offs involved when such optimizations are enabled or disabled.

One should note that test appliances can be either physical or virtual. If virtual test appliances are used, it is necessary that for any timing measurements such as latency, convergence time, etc. both test appliances should use the same clock reference. Currently, because of the delays introduced by the hypervisor, any mechanisms to implement timing synchronization may not be accurate. In such cases, alternate methodologies such as stateful traffic or TWAMP should be used to get accurate results. In case of physical test appliances this is not an issue as the current timing synchronization mechanisms such as GPS sync or IEEE 1588 [i.9] provide sufficient accuracy to the level of nanoseconds.

C.4 Benchmark Performance Metrics

At a high level, performance benchmarking metrics can be categorized in two ways - QoS based metrics and QoE based metrics.

C.4.1 QoS metrics

QoS metrics are the core network performance metrics based on network measurements to control the quality level of applications and services. Essentially the QoS metrics tell us the network impact on the quality of services.

Following are some of the key QoS metrics based on the performance forwarding benchmarking standards.

C.4.1.1 Throughput

Defined as the fastest rate at which the count of test frames transmitted by the DUT is equal to the number of test frames sent to it by the test equipment.

C.4.1.2 Latency

For store and forward devices, it is defined as the time interval starting when the last bit of the input frame reaches the input port and ending when the first bit of the output frame is seen on the output port.

For bit forwarding devices, it is defined as the time interval starting when the end of the first bit of the input frame reaches the input port and ending when the start of the first bit of the output frame is seen on the output port.

C.4.1.3 Frame Loss Rate

Defined as the percentage of frames that should have been forwarded by a network device under steady state (constant) load that were not forwarded due to lack of resources.

Frame loss Rate is calculated as *((input_count - output_count) * 100) / input_count.*

C.4.1.4 Back-to-Back

Defined as fixed length frames presented at a rate such that there is the minimum legal separation for a given medium between frames over a short to medium period of time, starting from an idle state.

The back-to-back value is the number of frames in the longest burst that the DUT will handle without the loss of any frames. The trial length should be at least 2 seconds and should be repeated at least 50 times with the average of the recorded values being reported.

C.4.1.5 Packet delay variation

It is defined as the variation of the packet latency when the packet traverses through the network. It is important to note that since the packet delay variation is a relative measure of delay experienced by the packets, the lack of clock synchronization of the testing endpoints may not impact the measurement but any drift or variation in the absolute value of the clock can impact the accuracy of this metric. It is recommended that, if possible, physical test devices are used for higher accuracy of the timing metrics till the industry resolves the timing synchronization of network endpoints in the virtual space.

C.4.1.6 Service Disruption Time for Fail-over Convergence

Fail-over convergence time is calculated as:

- Overall Convergence Time = Timestamp of 1st packet of last stream on backup - Timestamp of 1st trigger event
- Traffic Switchover Time = Timestamp of 1st packet of last stream on backup - Timestamp of 1st packet of first stream on backup
- Trigger event duration = Timestamp for last trigger event - Timestamp of first trigger event

Convergence and switchover tests consist of switching traffic from a primary path to a secondary path. The primary and secondary paths can be on 2 separate ports or 2 different paths on the same port (e.g. using VLANs or MPLS labels). It is required to detect that traffic has switched from primary path to secondary path and to measure service disruption time across 2 ports. It is also required to measure the convergence time from the control plane trigger event.

Ethernet based switching convergence (STP, ERPS) involves some initial flooding before the system recovers. It is important to account for duplicate and out of order frames when measuring the real frame loss to get an accurate convergence measurement.

One should note that, while packet loss can provide a measure of QoS, QoE involves the additional time till the service is fully recovered (defined as the time duration for continuous service availability without any disruptions or packet loss). Total service disruption time may consist of multiple disruptions before the service recovers completely.

C.4.2 QoE Metrics

Existing QoS metrics measure the network impact on the services and applications but fail to capture the subjective aspects associated with human experience. QoE metrics are specific to the type of service or application, as listed below:

- HTTP: page load time, load time variance.
- Video: MOS-AV score, range = 2-5 with 5 being the best.
- HTML5 video - AS score, 100 % score as the maximum.
- Audio & video conference applications - their traffic might consist of blend of protocols such as voice, video, text, FTP, etc. It is important to measure the QoE metrics for each internal protocol supported by them, but at the same time it is important to measure the synchronization across the multiple protocols. E.g. Video needs to be in synchronization with audio for a high QoE for the service.
- Direct metrics - Please note that some these metrics require full reference i.e. need the original signal for accurate measurement:
  - Peak Signal to Noise Ratio (PSNR) - Mean Square Error (MSE) between the original and the received image.
  - Structural Similarity (SSIM) - Combines luminance, contrast and structural similarity to compare the original image with the received image.
  - Video Quality Metric (VQM) - Blurring, global noise, block and color distortions.
  - Mean Opinion Score (MOS) - This metric combines delays, perceived jitter at application layer, codec used for communication and packet loss at application layer.
- Indirect metrics: These metrics are not related directly to the content but to the artifacts or side-effects that influence the perceived quality of the application or service:
  - Startup time: Time difference between sending the request for content and the time when the user actually received the content.
  - Delivery synchronization - In a multicast many-to-many scenario it is important that the content is received by all participants at the same time. Consider online gaming or video conferencing.
  - Freshness: The time difference between the time when the content is actually generated and the time when the users receives it, e.g. celebrating a goal with friends while watching a sports event.
  - Blocking: When the buffers on the receiver are empty and the user has to wait for content.

C.5 Additional Performance Metrics (white box testing)

Black box testing primarily focuses on the performance benchmarking of the functions/services supported by DUT. What mechanisms the DUT uses to support the given scalability or capacity is internal to the DUT's implementation and is not revealed to a large extent when black box testing is used.

Consider a case of a routing device under test. The parameters which we measure for scalability are RIB table size, FIB table size and for performance we measure route processing time, route installation rate and fail-over convergence time to support high availability. These can be measured with reasonable accuracy using black box testing.

There are some useful characteristics of a device under test that can be revealed if we measure the impact on CPU utilization, memory utilization and disk I/O utilization under varying type and varying level of load.

Different implementations use various mechanisms to optimize the HW resource utilization such as superior architectures, code design and queuing mechanisms. It is our understanding that the optimizations done in the implementations vary with each design and it is out of the scope for testing to measure the performance impact of varying load on the optimizations. Though, we do have to measure the effect of those optimizations which can be accurately measured by black box testing methods such as measuring RIB size in a router, or session rate for an access device.

Additionally, different types of load patterns (flat, sawtooth, burst, sinusoid, stair, random) and distributions (ABR, CBR, and VBR) stress the DUT in different ways. Though the metrics measured in each scenarios might be the same, the difference in traffic type provides more insight into DUT performance.

Annex D (informative): Performance evaluation of an IP edge data plane

This annex contains the performance evaluation of an IP edge data plane. This test is intended to characterize the throughput of a simplified version of a BRAS/BNG network function, both on bare metal and virtualised, thus providing a measurement of the capabilities of COTS HW to deal with data plane workloads on an intermediate node.

It is intended to address the following questions:

1) Data plane virtualisation. When running over standard server bare metal, what is the maximum throughput achievable?

2) Data plane virtualisation. When running over standard server bare metal, what are the bottlenecks or relevant HW features to take into account?

3) Data plane virtualisation. When running over an hypervisor, what is the performance degradation? What is the gap with respect to bare metal? What could be enhanced?

Basic BRAS features at the wire line network edge required in the fast path are implemented for this test, specifically Q-in-Q termination, MAC table management, and tunnelling encapsulation/de-encapsulation. Routing lookups have been also implemented.

Maximum throughput has been measured on both bare metal and virtualised, using qemu-kvm. Analysis on performance bottlenecks has been done for both scenarios as well.

This test is directly related to the BRAS/BNG network function and its virtualised version (vBRAS/vBNG). However, the architecture required to handle the load is quite general and can be applied to many other network functions as long as they support forwarding, some table management in the fast path, and per-subscriber encapsulation/de-capsulation. Thus, the features chosen in this use case are similar to other common network edge functions (e.g. CGNAT, P-GW, etc.)

It should be made clear that tests were run with two different processor generations. As will be highlighted, performance in bare metal is increased due to a higher number of processing capacity (higher number of cores). In the virtualised environment, while some bottlenecks appeared with the older processor generation, those bottlenecks were not present in the newer generation, and performance becomes similar in the virtualised and bare metal scenarios.

D.1 Detailed description of the System Under Test

The System Under Test is a simplified BRAS, which routes IP packets between a core network and CPEs (Customer Premises Equipment). IP Packets between CPEs and BRAS are encapsulated using QinQ, which is a typical encapsulation of an IPoE scenario, while IP packets between BRAS and core network are encapsulated using GRE and MPLS, which can be taken as a worst case situation for encapsulation in Internet access scenarios with wholesale support. The BRAS is, therefore, in charge of making the protocol conversion between QinQ and GRE.

Figure D.1 shows the internal structure of the testing BRAS:

- 2x 10 Gbps interfaces to the CPE side, typically a MAN
- 2x 10 Gbps interfaces to the core network side
- SW modules that run as threads in different processor cores:
    - Load balancer modules, responsible for load balancing per 10 Gbps interface. No hardware load balancing (aka RSS) is used. These load balancers will distribute the traffic to worker modules, responsible for the rest of the processing.
    - Worker modules, responsible for the QinQ encapsulation/de-encapsulation to/from the CPE side, MAC table management, routing, and tunnel encapsulation/de-encapsulation to/from the core network side.

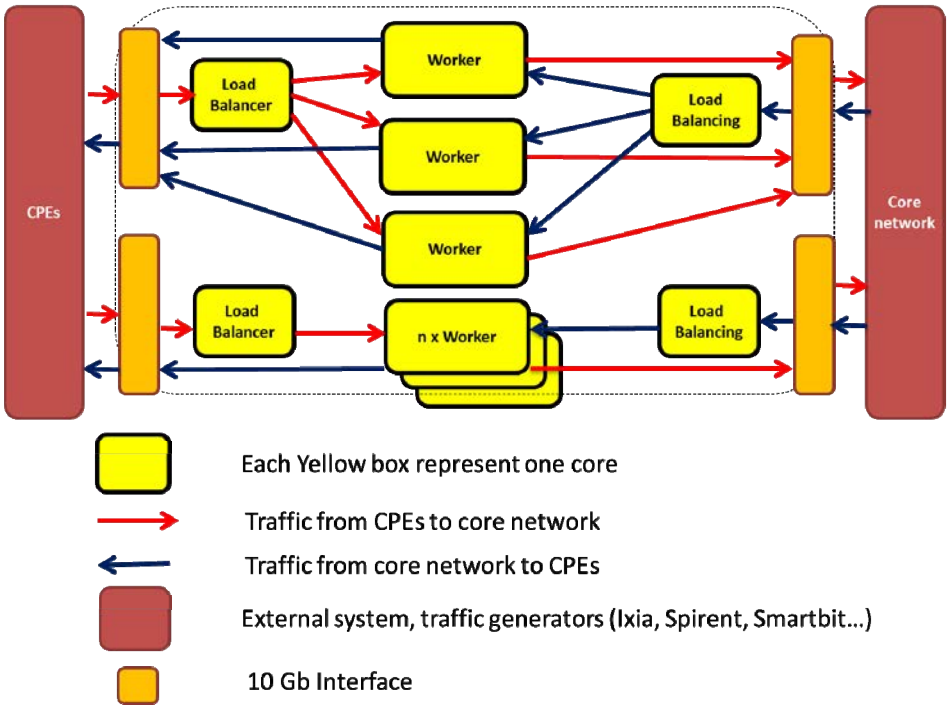


Figure D.1: Internal structure of the testing BRAS

Figure D.2 shows the internal structure of the Worker module.

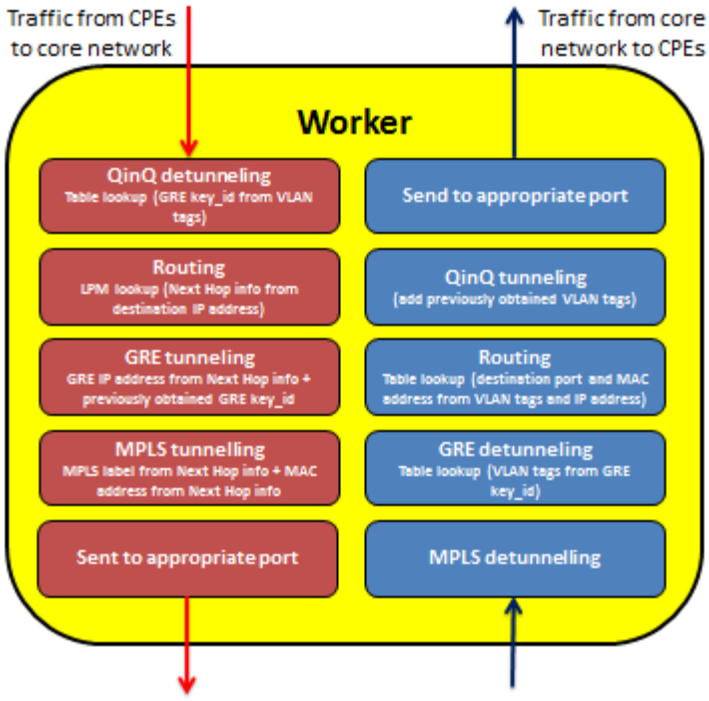


Figure D.2: Internal structure of the 'Worker' module

D.1.1 From CPE to core network

The BRAS receives QinQ packets, retrieves the QinQ tag and calculates the GRE id (based on a pre-loaded QinQ to GRE table). Then, it removes the QinQ tag and encapsulates the packet in a GRE header. It should be noted that the BRAS has being designed to support at least 32 768 CPEs per interface, i.e. **65 536 CPEs** in total. While there is a one-to-one mapping between the gre_id and the QinQ tags, there is no obvious formula to find one from the other: the mapping should come from lookup tables.

The BRAS stores the source MAC addresses from CPE based on the incoming QinQ tags and IP source address within ARP requests. Those MAC addresses will be used when sending packets towards CPE. If no ARP request is received within a configurable time (set to 1 500 seconds), the MAC entry is deleted.

The BRAS also does routing: it retrieves from the destination IP address the output port, the destination IP address of the GRE tunnel, the MAC destination address of the GRE tunnel, and its MPLS tag. This information is retrieved through Longest Prefix Match search in routing tables. Routing tables in the tests were populated with around **8 192 routes**, which is a typical route table sizing for an edge BNG/BRAS.

Figures D.3 and D.4 show the structure of the frames (only the relevant fields) before and after traversing the BRAS.

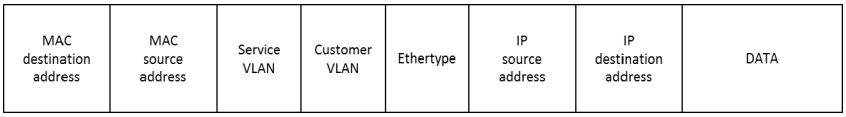


Figure D.3: Frame structure from CPE to BRAS

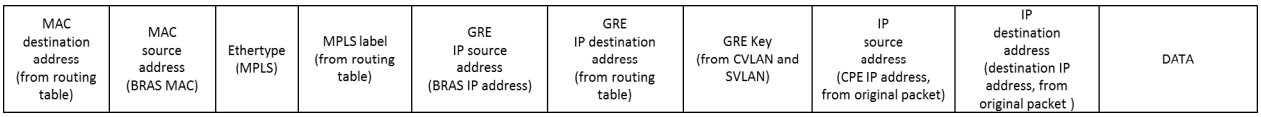


Figure D.4: Frame structure from BRAS and core network

D.1.2 From core network to CPE

The BRAS receives IP packets encapsulated in a GRE tunnel, together with MPLS tagging. It removes the MPLS tag and decapsulates the GRE packet; it retrieves the gre_id, and retrieves the QinQ tag (CVLAN and SVLAN) from the gre_id in the pre-loaded GRE to QinQ table. It removes the GRE headers and adds QinQ tagging.

Based on the gre_id (or QinQ tags) and destination address, the BRAS also retrieves the destination MAC address and destination port.

Figures D.5 and D.6 show the structure of the frames (only the relevant fields) before and after traversing the BRAS.

Figure D.5: Frame structure from core network to BRAS

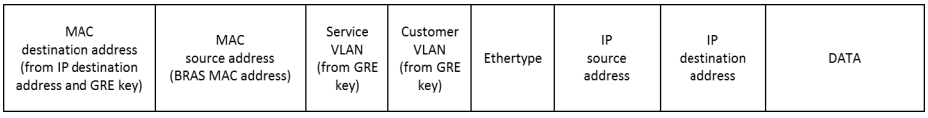


Figure D.6: Frame structure from BRAS to CPE

Figure D.7 shows the whole packet exchange process for both directions, indicating how the most relevant packet fields are converted when going through the BRAS.

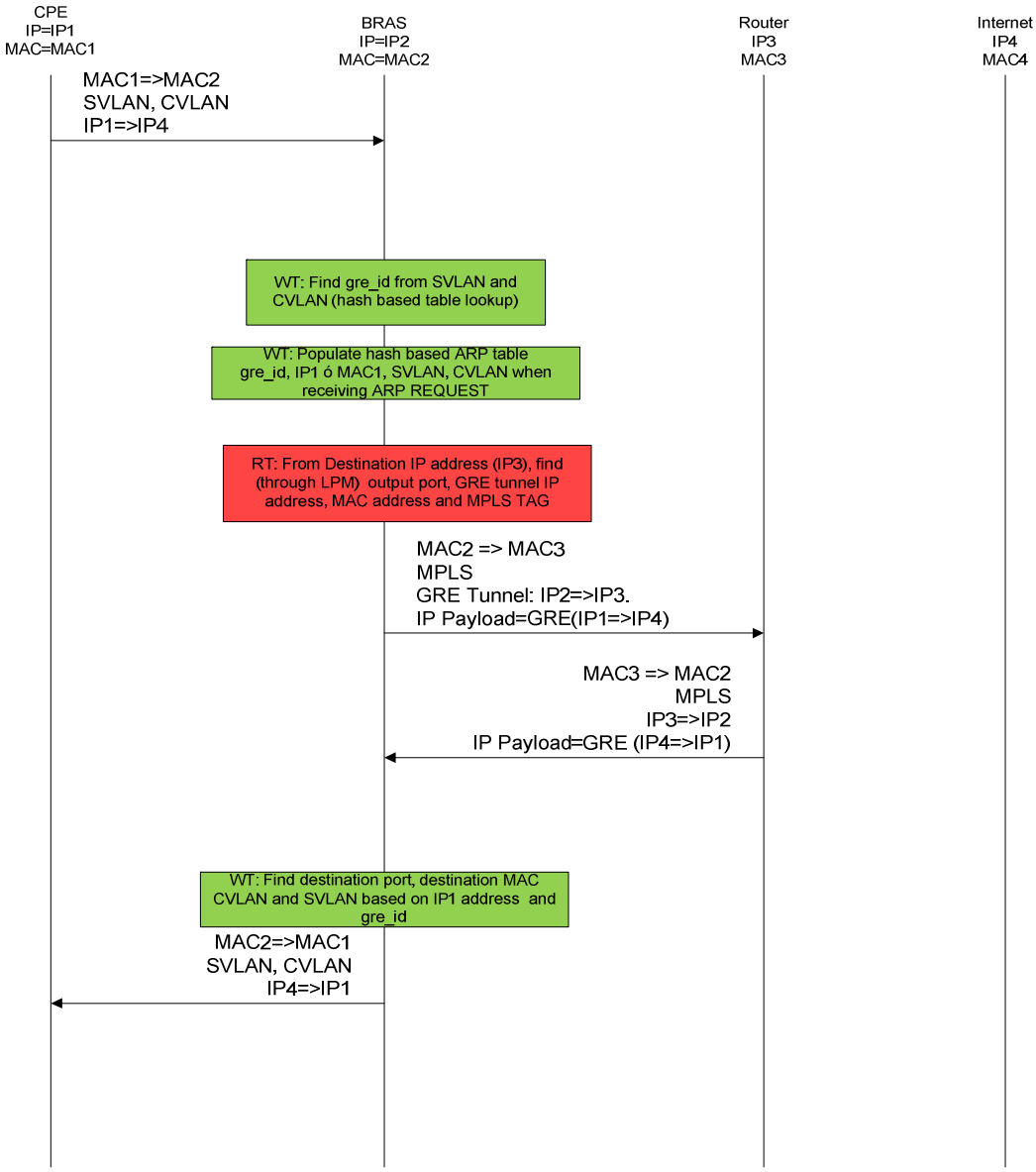


Figure D.7: Internal BRAS workflow and packet exchange diagram

D.1.3 Routing

The performance of the LPM routing lookups might depend on the routes configuration: the depth of each route (/24, /27, /16…), whether all routes are equally used or some routes are more used than others (resulting in better cache affinity), or even if some routes can share the same cache lines.

Hence, it is important to describe correctly the routes so that results between two configurations can be compared. As mentioned before, routing tables in the tests were populated with 8 192 routes, all being /24. Specifically, the routes are **0000101x.xxxxxxxx.xxxx0000.00000000/24** where x is randomly a '0' or a '1' (so from 10.0.0.0/24 to 11.255.240.0/24). The gap in the routes (the last four 0) should give a rather worst case scenario, where most of those routes will hit different cache lines (it is expected that routes like 10.0.0.0/24 and 10.0.1.0/24, 10.0.2.0/24, … could share the same cache line).

Traffic was generated equally distributed among those routes. IP destination addresses were generated randomly (not sequentially) by load generator, within the range of the routes.

D.2 Test environment

D.2.1 HW and SW used for the test with the older processor generation

Hardware Description

NOTE: This clause includes several trade names of products supplied by Intel®. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Item	Description	Notes
Platform	Intel® Server Board S2600CP Family	
Form factor	RMS 4U	
Processor(s)	2x Intel® Xeon® CPU E5-2690	20 480 KB Cache, with Hyper-threading enabled
Cores	8 physical cores/CPU	16 Hyper-threaded cores per CPU for 32 total cores
Memory	16 GB RAM (8x 2GB)	Quad channel 1333 DDR3
NICs	4x Intel® Niantic 82599 10 Gigabit Ethernet Controller	
BIOS	SE5C600.86B.01.02.0006.032920121341	Intel® Virtualisation Technology for Directed I/O (Intel® VT-d) enabled Turbo Boost was not enabled

Software Description

Item	Description	Version/Configuration
Host OS	CentOS	CentOS 6.2 • Kernel was updated to 3.2.27 • Unnecessary services disabled (e.g. firewall disabled, irqbalance service disabled)
Guest OS	CentOS	Used as the guest OS for virtualised configurations • Kernel was updated to 3.2.27 • Unnecessary services disabled (e.g. firewall disabled, irqbalance service disabled)
Virtualisation technology	Qemu-kvm	Version 1.3.0
IP stack acceleration	Intel® Data Plane Development Kit (Intel® DPDK)	Version 1.3.0_183
BRAS prototype application	Internal application based on DPDK to characterize the BRAS performance	Plans are afoot to make the source code publically available

D.2.2 HW and SW used for the tests with the newer processor generation

Hardware Description

NOTE: This clause includes several trade names of products supplied by Intel®. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Item	Description	Notes
Platform	Intel® Server R2308IP4LHPC (S2600IP) Family	
Form factor	RMS 2U	
Processor(s)	2x Intel® Xeon® CPU E5-2690 V2	25 MB Cache, with Hyper-threading enabled
Cores	10 physical cores/CPU	20 Hyper-threaded cores per CPU for 40 total cores
Memory	32 GB RAM (8x 4GB)	Quad channel 1333 DDR3
NICs	4x Intel® Niantic 82599 10 Gigabit Ethernet Controller	
BIOS	S2600IP_W2600CR_SFUP_BIOS01080003_BMC01170r4151_FRUSDR109_ME020107112	Intel® Virtualisation Technology for Directed I/O (Intel® VT-d) enabled Turbo Boost was not enabled

Software Description

Item	Description	Version/Configuration
Host OS	CentOS	CentOS 6.4 • Kernel was updated to 3.2.51 • Unnecessary services disabled (e.g. firewall disabled, irqbalance service disabled)
Guest OS	CentOS	CentOS 6.4 Used as the guest OS for virtualised configurations • Kernel was updated to 3.2.51 • Unnecessary services disabled (e.g. firewall disabled, irqbalance service disabled)
Virtualisation technology	Qemu-kvm	Qemu-kvm-0.12.1.2-2,355.0.1.el6.centos.3.x86_64 (as provided by CentOS)
Hugepages	Number and size of HugePages	24 * 1GB on Host and 2x 1G in VM
IP stack acceleration	Intel® Data Plane Development Kit (Intel® DPDK)	Version 1.4.1
BRAS prototype application	Internal application based on DPDK to characterize the BRAS performance	Plans are afoot to make the source code publically available

D.2.3 Test configuration

Figure D.8 shows the test configuration. Only the first CPU socket of the server and 4 × 10 GE interfaces were used for the tests.

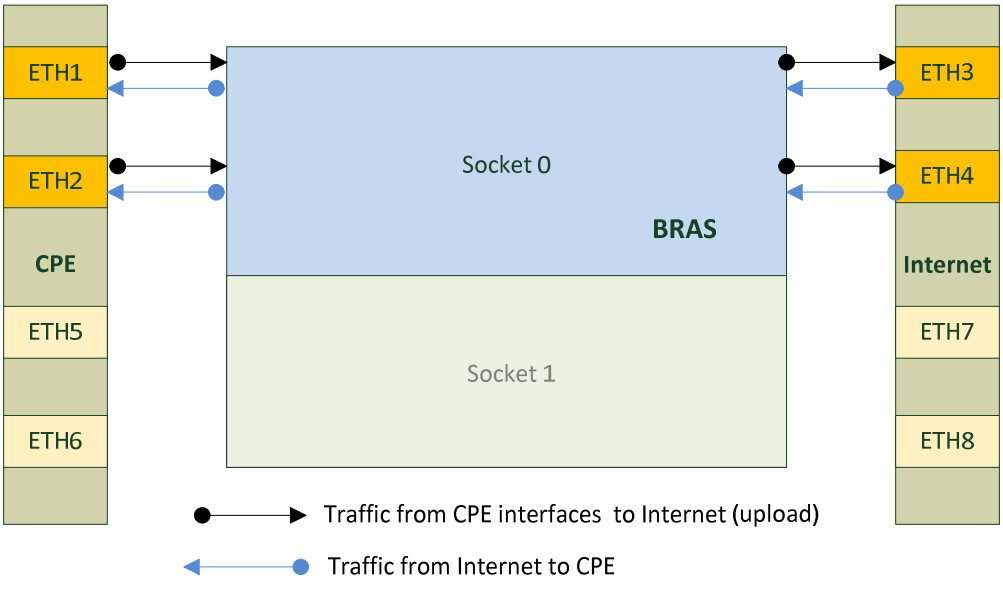


Figure D.8: Test configuration

The performance in the virtualised scenario is obtained using the system configured with only one VM which runs on CPU socket 0.

Measurements of throughput were made, using frame sizes from 64 bytes to 1 500 bytes. Performance figures are expressed in kilo packets per seconds.

D.3 Results

D.3.1 Results with the older processor generation

D.3.1.1 Bare Metal scenario

Running on bare metal, each 10G interface can handle **9 330 kpps**; so, the BRAS (running only on socket 0) can handle **37 Mpps** (around 18,6 Mpps from the CPE and 18,6 Mpps from core network). The performance is limited by the CPU: some cores are being used at 100 %. Maximum theoretical rate would be 12,755 Mpps per interface.

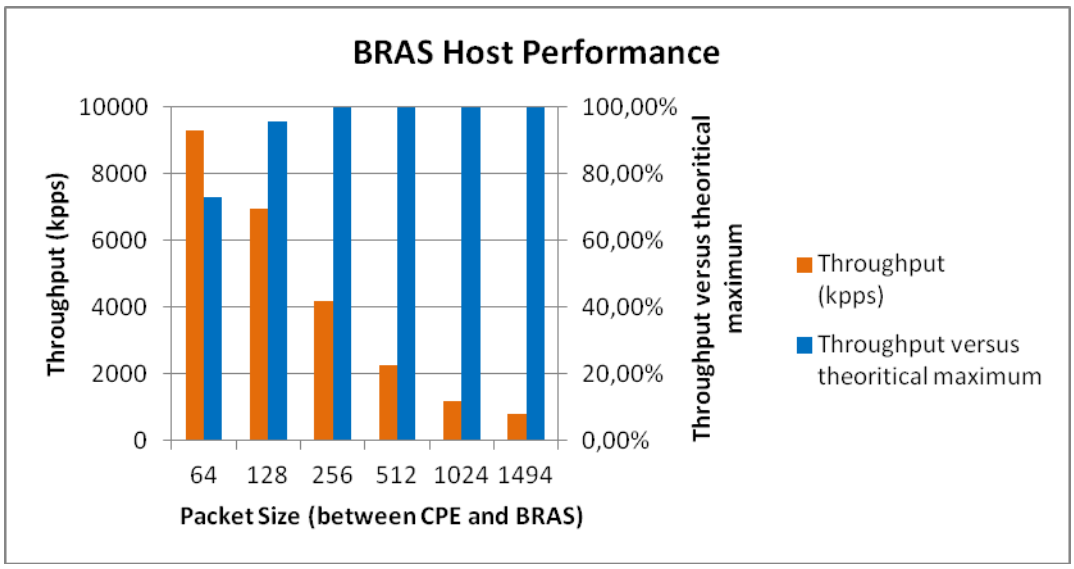


Figure D.9: Test results on bare metal scenario

It is expected that a full system, consisting of two BRAS, one per socket, each handling 4 interfaces, would be able to handle around **74 Mpps** (37 Mpps from the CPE and 37 Mpps from the core network).

D.3.1.2 Virtualised scenario

In a virtualised environment, each interface can handle **2 530 kpps;** so the BRAS (running only on socket 0) can handle around **10 Mpps** (5,1 Mpps from the CPE and 5,1 Mpps from core network). Maximum theoretical rate is still 12,755 Mpps per interface.

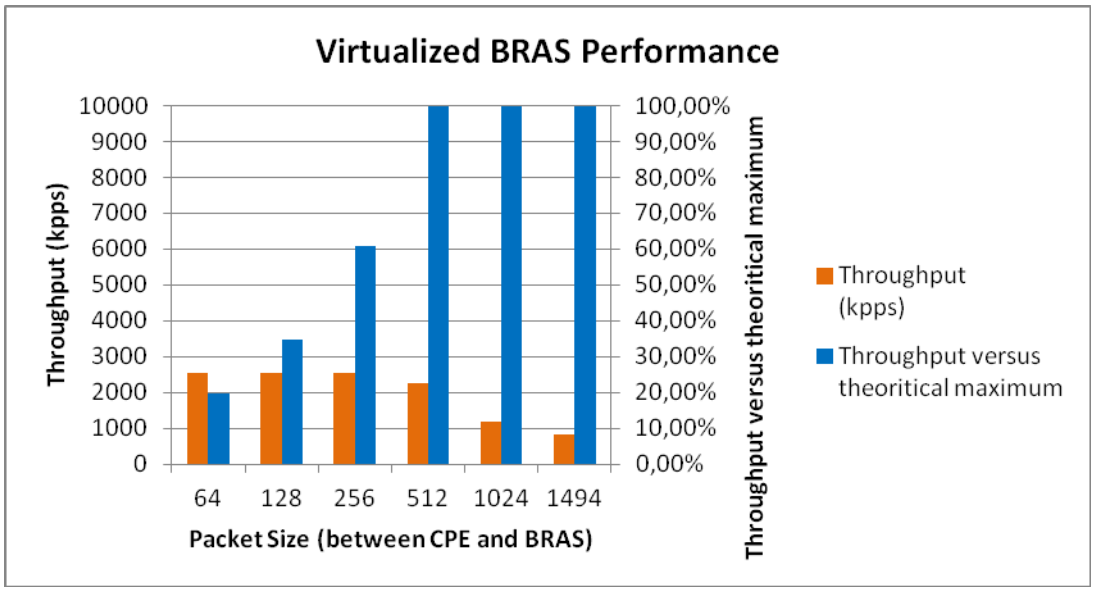


Figure D.10: Test results on virtualised scenario

The difference in the virtual environment, compare to bare metal one, is due to IOTLB eviction. On bare metal, large IO pages can be exploited; however, in a virtualised environment, this is not possible. While the current processors have HW support for memory page translation from process 'logical memory pages to VM' virtual memory pages and from these to host' physical memory pages, this support is restricted to small pages. Thus, it is not possible to rely on HW support for the page translation of large pages.

D.3.2 Results with the newer processor generation

D.3.2.1 Bare Metal scenario

Running on bare metal, each 10 G interface can handle **10 770 kpps**; so, the BRAS (running only on socket 0) can handle **43 Mpps** (around 21,5 Mpps from the CPE and 21,5 Mpps from core network). The performance seems to be limited by the PCI bus: cores are not being used at 100 %. Maximum theoretical rate would be 12,755 Mpps per interface.

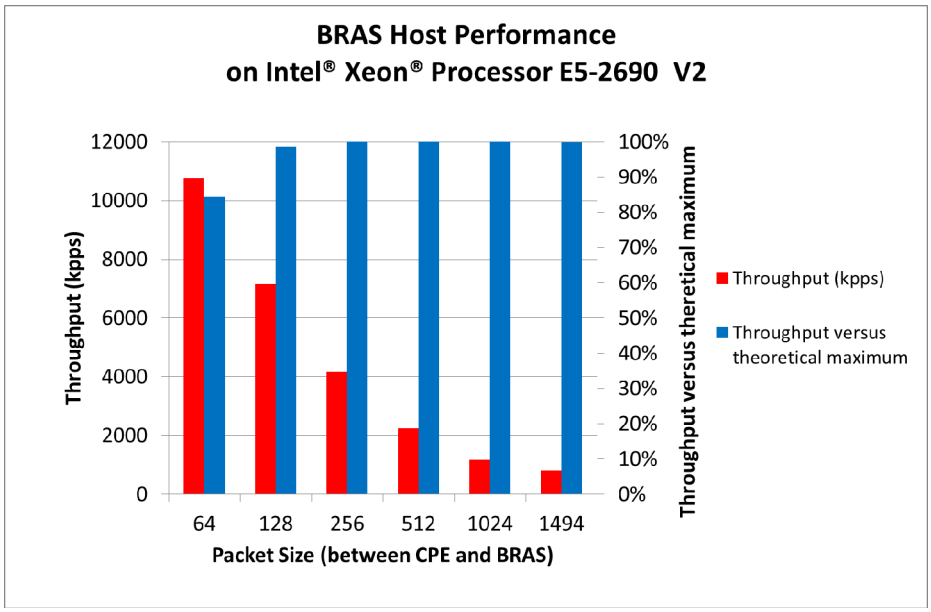


Figure D.11: Test results on bare metal scenario with the newer processor generation

It is expected that a full system, consisting of two BRAS, one per socket, each handling 4 interfaces, would be able to handle around **86 Mpps** (43 Mpps from the CPE and 43 Mpps from the core network).

The increase in throughput with respect to the bare metal scenario with the older processor is due to a higher number of processing capacity (higher number of cores dedicated to Worker modules).

D.3.2.2 Virtualised scenario

In a virtualised environment, each interface can handle the same amount of traffic than in the virtualised case (**10 750 kpps;** so the BRAS (running only on socket 0) can handle around **43 Mpps** (21,5 Mpps from the CPE and 21,5 Mpps from core network). Maximum theoretical rate is still 12,755 Mpps per interface.

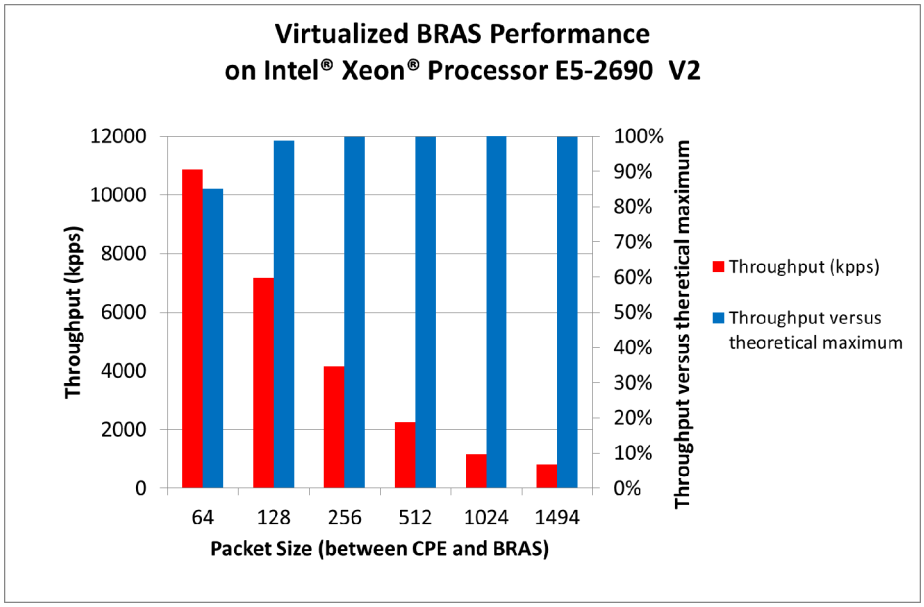


Figure D.12: Test results on virtualised scenario with the newer processor generation

The bottlenecks that appeared with the older processor generation (IOTLB eviction for large pages) have been removed in the newer generation, and performance becomes similar in the virtualised and bare metal scenarios, which show negligible differences.

Annex E (informative): Bibliography

IETF RFC 3393: "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)".

NOTE: Available at: http://www.ietf.org/rfc/rfc3393.txt.

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

ETSI GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework".

ETSI Directives.

NOTE: Available at: http://portal.etsi.org/Directives/home.asp.

History

Document history		
V1.1.1	June 2014	Publication