ETSI GS NFV-IFA 001 V1.1.1 (2015-12)

Network Functions Virtualisation (NFV); Acceleration Technologies; Report on Acceleration Technologies & Use Cases

Reference

DGS/NFV-IFA001

Keywords

NFV, acceleration, use case

Contents

Intellectual Property Rights

Foreword

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

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

The trademarks mentioned within the present document are given for the convenience of users of the present document and do not constitute an endorsement by ETSI of these products.

Modal verbs terminology

1 Scope

The present document provides an overview of NFV acceleration techniques and suggests a common architecture and abstraction layer, which allows deployment of various accelerators within the NFVI and facilitates interoperability between VNFs and accelerators. The present document also describes a set of use cases illustrating the usage of acceleration techniques in an NFV environment.

2 References

2.1 Normative references

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

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

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

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

Not applicable.

2.2 Informative references

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

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

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

[i.1] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.2] ETSI GS NFV-INF 003: "Network Functions Virtualisation (NFV); Infrastructure; Compute Domain".

[i.3] ETSI GS NFV-INF 005: "Network Functions Virtualisation (NFV); Infrastructure; Network Domain".

[i.4] ETSI GS NFV-IFA 002: "Network Functions Virtualisation (NFV); Acceleration Technologies; VNF Interfaces Specification".

3 Definitions and abbreviations

3.1 Definitions

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

para-virtualisation: virtualisation technique in which guest operating system virtual device drivers include software that works directly with specific hypervisor back-end interfaces for device access

NOTE: The virtual device interface is often similar to but not identical to the underlying hardware interface. The intent of para-virtualisation is to improve performance compared to the host fully emulating non-virtualised hardware interfaces.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in ETSI GS NFV 003 [i.1].

AAL	Acceleration Abstraction Layer
APU	Accelerated Processing Unit
ARP	Address Resolution Protocol
ASIC	Application-Specific Integrated Circuit
CoMP	Coordinated MultiPoint radio
CPU	Central Processing Unit
DOPFR	Dynamic Optimization of Packet Flow Routing
FPGA	Field-Programmable Gate Array
GENEVE	GEneric NEtwork Virtualisation Encapsulation
GPU	Graphic Processing Unit
HWA	Hardware Acceleration
IKE	Internet Key Exchange protocol
NFV	Network Functions Virtualisation
NFVI	NFV Infrastructure
NPU	Network Processor Unit
NV-DIMM	Non-Volatile Dual In-line Memory Module
NVGRE	Network Virtualisation using Generic Routing Encapsulation
NVMe	Non-Volatile Memory express™
OSPF	Open Shortest Path First
OVSDB	Open vSwitch® Database
RDMA	Remote Direct Memory Access
RIP	Routing Information Protocol
SoC	System on Chip
SRTP	Secure Streaming Real-time Protocol
TRILL	Transparent Interconnection of Lots of Links
vCPE	virtual Customer Premises Equipment
VNF	Virtualised Network Function
VPN	Virtual Private Network
VxLAN	Virtual extensible Local Area Network

4 Overview

4.1 General

The NFV Infrastructure (NFVI) includes the totality of all hardware and software components that build up the environment in which virtualised network functions (VNFs) are deployed. However, some VNFs may require some form of acceleration to be provided by the NFVI to meet their performance goals. While industry standard IT servers can support a large range of NFV use cases, some use cases are more challenging, especially relating to VNFs that need to meet certain latency or SLA requirements.

However, acceleration is not just about increasing performance. NFVI operators may seek different goals as far as acceleration is concerned:

- Reaching the desirable performance metric at a reasonable price.
- Best performance per processor core/cost/watt/square foot, whatever the absolute performance metric is.
- Reaching the maximum theoretical performance level.

NOTE: In this context, "Performance" can be expressed in throughput, packets per second, transactions per second, latency.

To allow multiple accelerators to co-exist within the same NFVI, and to be used by multiple virtualised network function components (VNFCs), several virtualisation technologies exist in the industry and they will continue to evolve. In general an acceleration abstraction layer (AAL) is used to aid portability of application software (see figure 1). The role of an AAL is to present a common interface for use by a VNFC, independent of the underlying accelerators. Different implementations of the AAL and bindings to different accelerator implementations can be provided without requiring changes to the independent VNFC code. All code which is dependent on the accelerators is within the AAL.

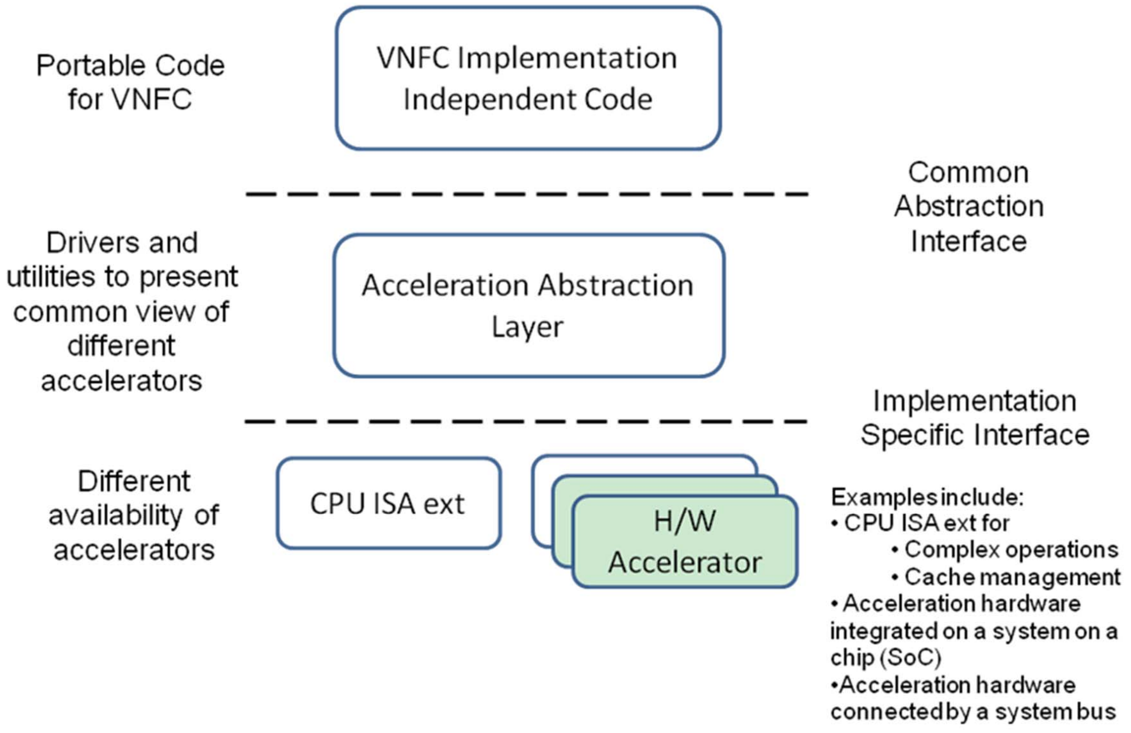

Portable Code for VNFC
VNFC Implementation Independent Code
Common Abstraction Interface
Drivers and utilities to present common view of different accelerators
Acceleration Abstraction Layer
Implementation Specific Interface
Different availability of accelerators
CPU ISA ext
H/W Accelerator
Examples include:
• CPU ISA ext for
• Complex operations
• Cache management
• Acceleration hardware integrated on a system on a chip (SoC)
•Acceleration hardware connected by a system bus

Figure 1: Use of acceleration abstraction layer (AAL) to enable fully portable VNFC code across servers with different accelerators

This AAL is a normal feature of operating systems and is normally implemented using a common driver model and hardware specific drivers. In the NFVI, the virtualisation layer in charge of compute and storage resources is typically implemented in the form of a hypervisor which plays the role of a base operating system which interfaces to the hardware. The hypervisor then provides common and uniform virtual hardware to all virtual machines so that VNFC code is fully portable.

In order to achieve full portability of VNFC code, the AAL can be entirely contained in the hypervisor. In this way, the virtualised accelerator presented to the VNFC is generic so that the host operating system of the VNFC can use generic drivers, without requiring awareness of the AAL.

However the performance of fully independent VNFCs may be less than desired because the hypervisor needs to emulate real hardware, so an alternate model known as para-virtualisation also exists. With para-virtualisation, AAL code is also present in the VNFC and is adapted to specific virtualisation drivers and hardware.

It is the intention of the present document to define and promote acceleration architectures that aid portability with the use of an AAL in the guest, host or both. The specification of an AAL when other forms of virtualisation are used and acceleration without use of an AAL are outside the scope of the present document. The present document does not intend to preclude any specific acceleration architectures from VNF deployments.

NFV Acceleration can be done by hardware, software or any combination thereof. The AAL should not prescribe a specific hardware or software implementation, but enable a spectrum of different approaches (including pure software).

4.2 Hardware Acceleration

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

These functions may be correlated to the three NFVI domains and subsequently address Compute, Network and Storage Acceleration. By using the term "functions", the present document abstracts the actual physical implementation of the hardware accelerator.

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

With AAL, multiple hardware accelerators can be presented as one common and uniform virtualised accelerator to the accelerating function and thus can work simultaneously for that function.

4.3 Software Acceleration

In addition to the rich selection of hardware acceleration solutions, modern, high performance CPU (as well as GPU or APU) silicon enables an alternative acceleration option - software accelerations.

Software acceleration provides a set of one or more optional software layers that are selectively added within elements of an NFV deployment (e.g. Compute, Hypervisor, VNF, etc.) to augment or bypass native software within a solution. Together, these new layers bring improved capabilities (e.g. increased network throughput, reduced operating overhead) which result in measurable improvements over standard, un-accelerated implementations. Software acceleration frameworks and software accelerators are the two major components built upon these layers to constitute a complete software acceleration solution.

There are several well-known software acceleration frameworks; one is Data Plane Development Kit (DPDK®). DPDK® works hand in hand with an underlying Linux operating system to "revector" network traffic outside of the Linux kernel and into user space processes where the traffic can be handled with reduced system overhead. When deployed appropriately into a virtual switch, this capability enables performance improvements over a native (un-accelerated) virtual switch. Additional improvements can be seen when elements of this open framework are implemented and deployed within a suitable VNF. Together, the combined acceleration results can be greater than either alone.

Another acceleration framework example is OpenDataPlane (ODP®) from the Linaro Networking Group. ODP® is an open source project which provides an application programming environment for data plane applications. ODP® offers high performance and portability across networking Systems on Chip solutions (SoCs) of various instruction sets and architectures. The environment consists of common APIs, configuration files, services, and utilities on top of an implementation optimized for the underlying hardware. ODP® cleanly separates its API from the underlying hardware architecture, and is designed to support implementations ranging from pure software to those that deeply exploit underlying hardware co-processing and acceleration features present in most modern networking "Systems on Chip" (SoCs) solutions.

Software accelerators are components which are typically (though not necessarily exclusively) built against corresponding software acceleration frameworks such as DPDK® and ODP®. Examples of such accelerators are Poll Mode Drivers that would utilize DPDK® fast path, or similar fast path mechanism built with ODP® APIs. When dealing with the concept of acceleration abstraction layer (AAL) with regard to software acceleration, it should be noted that AAL provides a common abstraction to a set of variant software accelerators, not a set of different software acceleration frameworks.

4.4 Heterogeneous Acceleration

4.4.1 General

Heterogeneous accelerators are another class of accelerated functions called from the VNFC (and differentiated from hardware accelerators described in clause 7.2.3 of ETSI GS NFV-INF 003 [i.2]). It refers to functions implemented within the compute node on the NIC, CPU Complex, accelerator blades / chassis, a plug-in card or an attached device such as FPGA, ASIC, NPU, and called from the VNFC, possibly on a fine granularity.

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

Implementation of heterogeneous accelerators may vary from vendor to vendor.

4.4.2 Coherent acceleration

4.4.2.1 Nature

Coherent hardware acceleration denotes a special execution context of acceleration where the accelerator and the CPU are closely coupled so that general memory (physical addressable or VNF virtual private address space) can be addressed directly from the accelerator. Coherent accelerator access can be done through new instructions available in the processor or through a special controlling interface in the processor.

The execution of accelerated function in the hardware may be synchronous or asynchronous to the CPU program. When asynchronous, the CPU or the controlling interface provides mechanisms for either notification (via interrupts or other mechanisms) or polling for the completion of the instruction execution.

The acceleration hardware may be on the same chip as the processor or elsewhere, connected through standard interfaces or private interfaces.

4.4.2.2 Runtime definable acceleration

Some acceleration hardware can be configured or programmed at runtime in such a way that the hardware does not define a specific acceleration function but is rather programmed/configured at runtime.

Runtime definable acceleration combines:

- Programmable/Configurable hardware such as FPGA, GPU, NPU, SoC or an extendable processor (instruction extension by microcode update for instance);
- "Firmware" for the hardware;
- Software that VNF can leverage to make use of the programmed/configured hardware.

The programming or configuration of the acceleration hardware is hardware specific, is done at Compute Node initialization so that VIM inventory is updated with created accelerators.

4.5 Classification of accelerators

4.5.1 General

As shown on figure 2, hardware, software and heterogeneous accelerators can be classified according to different criteria or multiple facets such as:

- what software would be making use of the accelerator [NFV Software];
- type of the accelerator [Type of accelerator];
- location of the accelerator [Housing/Location];
- functionality type.

NOTE: Figures 2, 3 and 4 are mostly driving the use cases described in the present document, hence are not the exhaustive list of accelerator taxonomy.

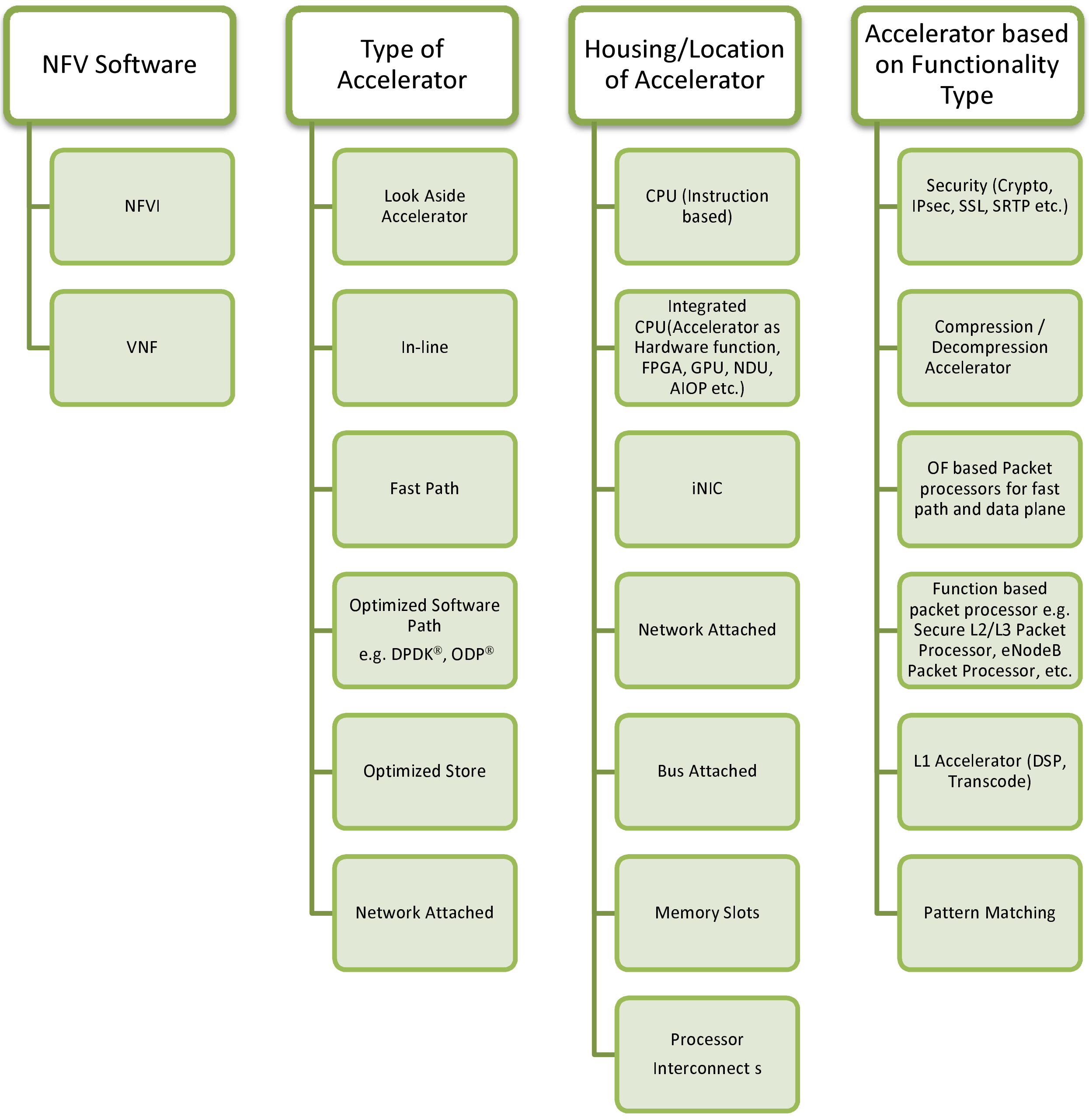

NFV Software
NFVI
VNF
Type of Accelerator
Look Aside Accelerator
In-line
Fast Path
Optimized Software Path e.g. DPDK®, ODP®
Optimized Store
Network Attached
Housing/Location of Accelerator
CPU (Instruction based)
Integrated CPU(Accelerator as Hardware function, FPGA, GPU, NDU, AIOP etc.)
iNIC
Network Attached
Bus Attached
Memory Slots
Processor Interconnect s
Accelerator based on Functionality Type
Security (Crypto, IPsec, SSL, SRTP etc.)
Compression / Decompression Accelerator
OF based Packet processors for fast path and data plane
Function based packet processor e.g. Secure L2/L3 Packet Processor, eNodeB Packet Processor, etc.
L1 Accelerator (DSP, Transcode)
Pattern Matching

Figure 2: Classification of accelerators

4.5.2 NFV Software

This classification is based on the possible NFV software candidates, that can make use of the accelerator. In the NFV architectural framework, there are two possible candidates namely:

1) NFVI:

This refers to the case, where a piece of software that is part of the NFVI, makes use of the accelerator (e.g. vSwitch).

2) VNFs:

This refers to the case where a piece of software within a VNFC code makes use of the accelerator.

NOTE: When NFVI software components are deployed as VNFs, those components would be under a separate administrative domain, providing an infrastructure for a separate NFVI domain from the one in which they are deployed.

4.5.3 Types of Accelerator

The classification is based on the possible types of accelerators, including:

1) Look-aside accelerator:

Accelerators of this type are typically algorithmic accelerators that are used to speed up compute intensive operations. These accelerators work typically in command/response mode, where the software submits the command and data to the accelerator. The accelerator processes the data based on the command and returns a response. Examples include crypto, protocol accelerators, pattern matching and compression.

2) In-line:

Accelerators of this type work in-line with software for packet processing.

3) Fast Path:

This refers to accelerators where the packet processing happens in a cut-through fashion without reaching the Host CPU.

4) Optimized Software Path:

In this case, the accelerator is an optimized software path. Examples include accelerators created using DPDK® or ODP® frameworks.

5) Optimized Store:

In this case, the accelerator function is an optimized store - e.g. NV-DIMM, Flash DIMM.

4.5.4 Housing/Location of Accelerator

This classification is done based on where the accelerator is housed located or realized. This classification includes:

1) CPU Instruction based:

In this case, the accelerator function is part of processor instruction set.

2) Integrated CPU:

In this case the accelerator is housed as a hardware function, (e.g. FPGA, GPU, NDU, AIOP) within the CPU socket.

3) iNIC:

In this case, the accelerator in this case is housed as part of iNIC.

4) Network Attached:

The accelerator is accessible through the network.

5) Bus Attached:

The accelerator functionality is accessible through a bus.

6) Memory Slots:

Memory device provides the accelerated function.

7) Processor Interconnects:

The accelerator is attached to the processor interconnect (which is a processor dependent feature).

4.5.5 Accelerator based on Functionality Type

This classification of accelerators is based on the actual functionality accomplished by the accelerator. It includes:

1) security (Crypto accelerator, IPsec, SSL, SRTP, etc.);

2) compression / decompression accelerator;

3) packet processors for fast path and data plane;

4) function based packet processor, e.g. Secure L2/L3 Packet Processor, eNodeB Packet Processor, etc.;

5) L1 Accelerator (e.g. DSP, Transcode);

6) pattern matching (e.g. DPI).

4.6 Accelerator Usage Models

4.6.1 General

As shown below there are two use cases, one for NFVI and the type of accelerators it may use (see clause 4.6.2) and the other for VNFs and the type of accelerator they may use (see clause 4.6.3).

4.6.2 NFVI Accelerator Usage

In this case, the accelerator is intended to improve the NFVI performance, so that the VNF can see a resulting performance gain (see figure 3). For example:

- A gateway VNF can benefit from a direct connected iNIC accelerator bypassing the virtualisation layer and hence achieve better performance;
- An VNF requiring a load balancer function can delegate the actual load balancing functionality to a vSwitch or vRouter of the NFVI instead of relying on a load balancer VNFC; or
- VNF instantiation can be accelerated by NFVI leveraging network and storage acceleration.

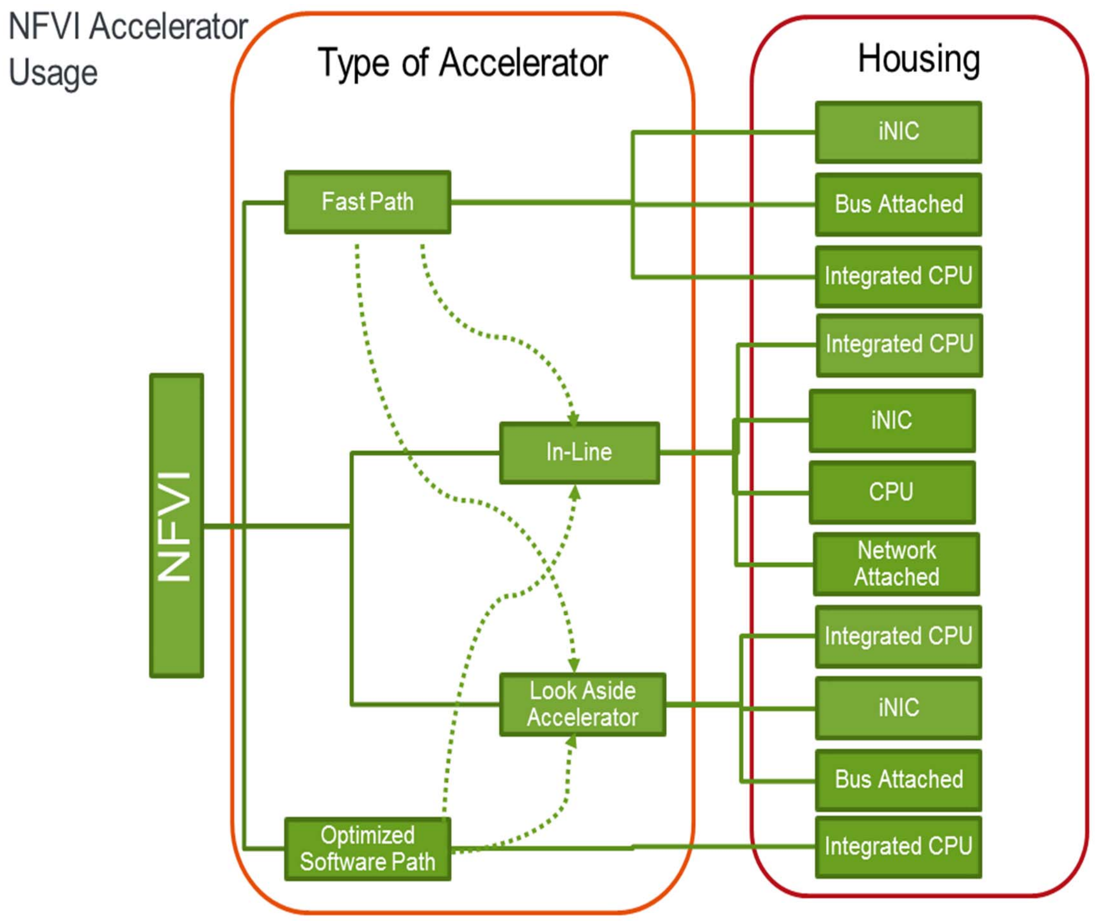


Figure 3: NFVI usage of accelerators

The NFVI may use:

- Look Aside Accelerator; which can be housed as part of:
  - Integrated CPU, iNIC or Bus Attached.
- In-line accelerator; which can be housed as part of:
  - Integrated CPU, iNIC, Network Attached or CPU.
- Fast Path; which can be housed as part of:
  - iNIC, Bus Attached or Integrated CPU.
- Optimized Software Path; which can be housed as part of:
  - Integrated CPU.

NOTE: Fast Path and Optimized Software Path may make use of Look Aside Accelerators or In-Line accelerators.

4.6.3 VNF Accelerator Usage

Some VNFs require accelerators to offload some portions of their packet processing or most of the data processing to meet their performance needs (see figure 4).

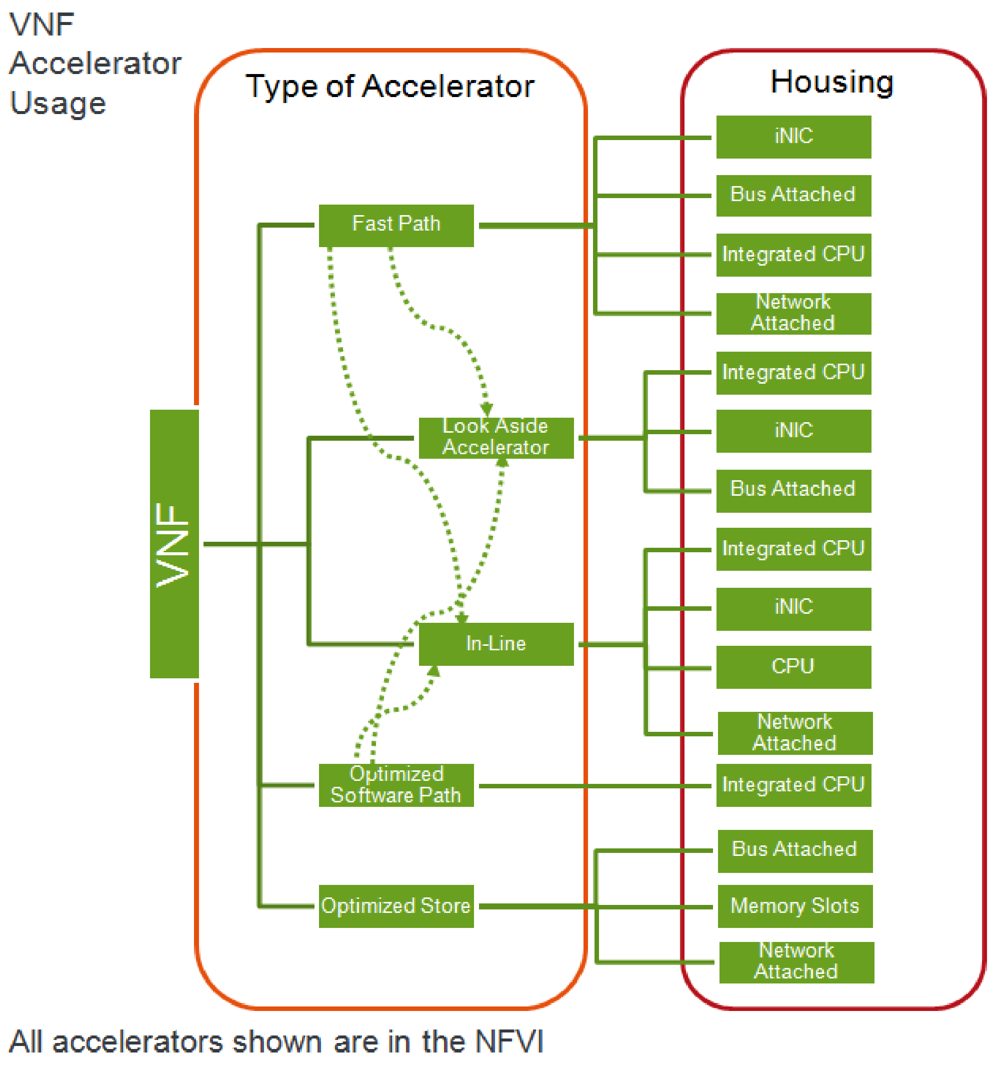


Figure 4: VNF usage of accelerators

The VNF may use:

- Look Aside Accelerator with housing in:
    - integrated CPU, iNIC or Bus Attached.
- In-line with housing in:
    - integrated CPU, iNIC, CPU or Network Attached.
- Fast Path with housing in:
    - iNIC or Bus Attached or Integrated CPU or Network Attached.
- Optimized Software Path with housing in:
    - integrated CPU.
- Optimized Store with housing:
    - on Memory Slots, Bus Attached or Network Attached.

NOTE: Fast Path and Optimized Software Path may make use of Look Aside Accelerator or In-Line Accelerator.

These categorizations of accelerators help in mapping individual application performance requirements and design to specific accelerator requirements.

5 Use Cases

5.1 Compute Acceleration

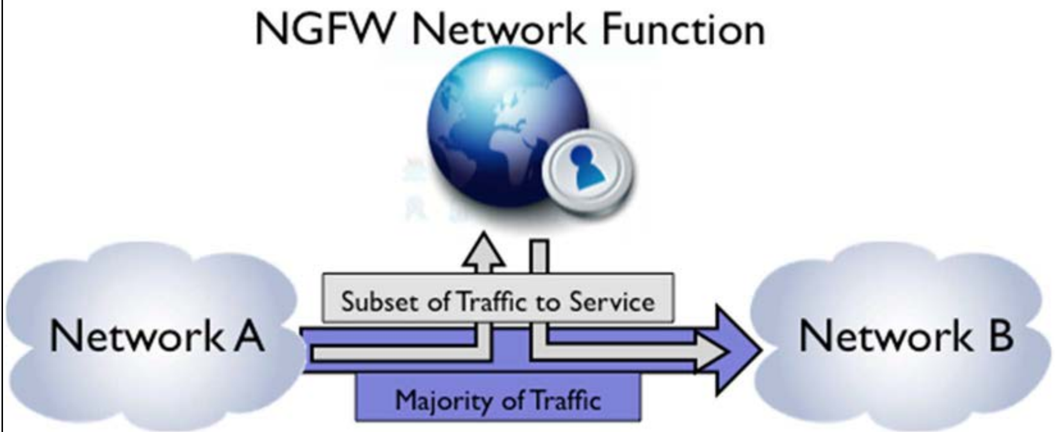
5.1.1 IPSec tunnels termination

Title *	IPSec tunnels termination VNFC
NFV Components *	VNFC VNFD VIM Orchestrator
Introduction & Problem Statement *	The Virtual Application is a router that terminates IPSec tunnels on one set of vNICs and routes traffic to another set of vNICs: • WiFi hot spot aggregation (tens of thousands of tunnels) • enodeB backhauling (thousands of tunnels) • Enterprise connectivity aggregation (hundreds of tunnels) • vCPE aggregation (tens of thousands of tunnels)
Performance Consideration *	a) IKE tunnel creation rate (in case a region of hot spots reconnects after an outage). b) Concurrent number of tunnelled interfaces is a key parameter. c) bandwidth or packets per second. d) Fragment handling (relevant for IPSec termination of the NIC)
Management & Orchestration Consideration	**Nf-Vi - Instantiation and VNFD** Contain a list needed crypto (RSA, AES-2048, etc.) and hash (MD5, SHA1, etc.) algorithms; if the list is mandatory (instantiation impossible if not there) or optional. That said, it may be wise to define a set of reachable performance for the VNFC and a required acceleration support: • 10 Gbps: 1 core, no hardware • 20 Gbps: 2 cores, AVX instruction set • 40 Gbps : 1 core, IPSec AES-2048 hardware support **VIM** The VIM maintains the list of available acceleration resources and their consumption by VNFs so that orchestration can properly identify instantiation targets for new VNFs.
Possible Accelerators	Accelerator types: look-aside, in-line, fast path, software. Accelerator locations: CPU (instruction set, native or with FPGA support), integrated in CPU, iNIC, bus attached. VNF leveraging of accelerators need to be independent from accelerator types and locations.

Description *	**Vn-Nf - VNF interface** (see note) The VNF needs to deal with IKE and dataplane aspects of IPSec. **Dataplane Crypto card** Application (Linux/DPDK®) reads/writes packets from NIC, application leverages the crypto interface to apply crypto operation, application routes the data. The interface is based on a standalone virtual, or more accurately, a synthetic device that will provide crypto operations interfaces much like virtio-net provides packet interface. **Data plane IPSec termination on a NIC** Application (Linux/DPDK®) reads(decrypted)/writes(to be encrypted) packets from NIC, application routes the data. The interface is the NIC but there is a need of a NIC IPSec Tunnel termination capability in the VNFD Information Element. Standard vNICs such as virtio-net and vmxnet3 need to be extended to signal capability and activate it. **Software library for CPU** Application (Linux/DPDK®) read/ and writes packets from NIC, executes crypto operation, application routes the data. The virtual application uses the library independently from NFVI. Instantiation needs VNFD description of achievable performance metrics depending on the processor available.
Other Considerations	**Live migration** Accelerators maintains contextual information (in particular if the NIC terminates the tunnels) that have to be migrated to a new system. Live migration from one hardware accelerator model to another one seems fairly impossible if any state is maintained. **Security** The provisions to avoid key leaks between VNFCs if the IPSec termination is fully offloaded to a single device are not addressed in the present document. The discussion on the maintenance of the FIPS compliance has not be addressed in the present document.
NOTE: For this architectural use case, the analysis is limited at high level and does not deal with the details of asynchronous behaviour of hardware chips, this will be detailed in ETSI NFV-IFA 002 [i.4]. Legend: * identify mandatory fields.	

5.1.2 Next Generation Fire Wall (NGFW) Acceleration

Title	Next Generation Fire Wall (NGFW). NGFW combines the functions of a standard firewall, Intrusion Prevention Systems (IPS), SSL VPN and Deep Packet Inspection (DPI) capability associated with user-ID and/or application-ID.
NFV Components	NFVI, Software Architecture, Performance and Security
Introduction & Problem Statement	Example required features of NGFW include the following: • Support for inline/passive/tap modes • Switched and routed network architectures • Layer 2 forwarding: bridge / switch Ethernet • Layer 3 forwarding: route IPv4/v6 packets • Network Address Port Translation (NAPT) • High-availability support • IPSec VPN termination/origination • Packet classification/filtering • Load balancing to host (x86) applications • Stateful flow processing • Zero copy delivery to host OS user mode applications • Cryptography support

Description	 NGFW Network Function Subset of Traffic to Service Network A Network B Majority of Traffic **Figure 5** With NGFW, customers / end-users can enforce user / application - specific policies, such as real-time protection against threats, by leveraging the performance of the underlying hardware, and its required stateful flow management capability – As a result it can manage the state and actions associated with millions of flows, dynamically at > 100 Gbps.
Functional & Performance	NGFW is a perfect example of an application that requires acceleration of a COTS platform, the hypervisor, and in some cases the underlying network infrastructure (see note). The result can be CPU-bound or I/O bound or both and can affect the underlying implementation. Performance requirement includes support for 100 GbE full-duplex. An example workload consists of the following: **Data plane - Key Functions:** • L2 forwarding and virtual bridges. • L3 routing and virtual routing. • IP VPN termination. • Network address and port translation (NAPT) - multiple modes of operation. • Stateful network flow analysis. • Packet classification and filtering. • Dynamic per flow application policy. • Load balancing to x86 or ARM on egress interfaces. • SSL identification. • SSL inspection. **Application plane on Host processor (x86 or ARM):** • DPI for application and protocol identification. • Snort for threat management. • Passive network discovery. • Targeted vulnerability assessment. **Control plane on Host processor (x86 or ARM):** • ARP. • OSPF. • RIP. • Application management GUI.
Possible Accelerators	NFVI Fast Path, In-Line, and Look-Aside Accelerators - all as iNIC, Bus Attached, and Integrated CPU - can all be used for acceleration in this use case.
Management & Orchestration Consideration	Managing the NGFW through a local or remote API is required. This acceleration is transparent to the VNFs. It is desirable that the orchestrator is able to discover the underlying performance capability of the compute node.
Related Use Case(s)	Load balancing, NAT and value-add Use Cases.
NOTE: The functional partitioning is workload specific and will depend on the IPS and / or Firewall application.	

5.1.3 Virtual Base Station (VBS) L1 Acceleration

Virtualisation and centralization of the Base Station resources (Cloud-RAN topology) at different scale can leverage resource utilization for load balancing among different base stations to provide cost reduction, high resource and spectrum utilization and energy efficient networks.

The main challenge in virtualising the base station is its compute-intensive baseband functions such as the PHY layer (L1), typically implemented on dedicated modern hardware/processors or on general purpose L1 hardware accelerators.

In this use case of virtual base station L1 Acceleration, the L1 software is virtualised and running as VM on high volume servers, leveraging a network attached or look aside hardware accelerator, covering several key challenges and architectures such as compute-intensive, real time processing and networking, abstraction layer between the software and acceleration resources and topologies for the physical and logical connectivity between CPU and acceleration resources.

The virtual Base Station include additional layers such as L2, L3, RRM and OAM, which also include certain compute-intensive functions that can be accelerated in a similar manners at the same or different hardware accelerators topologies. For example, the accelerators can be inserted or integrated in CPU blades / chassis via high bandwidth interface, such as rapid I/O, PCIe, and Ethernet. While the pure implementation of the accelerated data transportation in this topology is simpler due to the accelerated data chain is CPU->accelerator->CPU generally, pooling the HWA resources out of server, provide higher level of flexibility, lifecycle, thermal efficiency and other cost benefits, as detailed in this use case.

Title	Virtual Base Station (VBS) L1 Acceleration Use Case
NFV Components	• VNF / VNFC. • VIM. • Compute Nodes. • Networking Nodes. • HWA (Hardware Acceleration) Nodes.
Introduction & Problem Statement *	**Introduction:** Virtualisation of mobile base station is expected to provide advantages such as lower footprint and energy consumption coming from dynamic resource allocation and traffic load balancing, easier management and operation, faster time-to-market and enablement of advanced algorithm for spectrum efficiency (e.g. CoMP). **Challenges:** The main challenge in virtualising the base station is its baseband PHY (L1) layer, which includes the most computational intensive signal processing tasks, such as channel coding/decoding, FFT/iFFT. Typically, those functions are implemented on dedicated modem processors or on general purpose L1 accelerators. **Virtualisation of HWA resources:** General purpose L1 modem processing unit accelerators contain processing elements blocks (Channel coding/decoding, FFT/iFFT, etc.), in which through combination of related blocks and service chaining of dataflow, accelerate the baseband software and enables the virtualisation of independent hardware acceleration resources similar to the virtualisation of CPU resources (cores, memory, I/O). **Abstraction Layer:** An abstraction layer such as modem programing language can be implemented between the L1 Hardware Acceleration (HWA) and the baseband software (CPU) virtual machines. Similar to the case of OpenCL and GPUs, an open abstraction layer would simplify the programing of the workload to be executed on the HWA, it would enable certain portability capabilities and a creation of open eco-system of hardware and software technology providers, VNF product/solution providers and Carriers. **Physical interface of HWAs and CPUs:** The physical interface between the CPU in which the baseband SW is running on and the L1 Hardware Acceleration (HWA) might be in a form of server attached card or module, for example a PCIe card over PCIe bus and the use of SR-IOV to share the acceleration resources between different baseband virtual machines on the specific server. While functionality wise, such configuration can work, it would not provide the same level of agility, flexibility, scalability, functionality and cost reduction (both capital and operational) as in the case of disaggregation of all resources model with a pool of HWA resources.

Introduction & Problem Statement *(continued)	**The disaggregation model description:** In the case of disaggregation model, the independent CPU blades / chassis are separated from the independent accelerators blades / chassis and connected by network elements. A preferred network protocol would be similar to the CPUs / Servers interconnect network (for example, Ethernet, Infiniband, PCIe, etc.) to allow either dedicated network or integration onto single network for all compute and acceleration resources. In order to support the low latency requirement between L1 Acceleration and baseband software the use of technologies such as SR-IOV, DPDK® or RDMA/RoCE can be used. **The disaggregation model benefits:** The advantages of the disaggregation model between CPU blades / chassis and Acceleration blades / chassis are: • Sharing the accelerators resources across many computing blades / servers. • Providing improved amortization of traditionally expensive components. • Independant upgrading and scaling of resources. • Increasing lifespan of each resource by enabling easy replacement of specific resource and allowing thermal efficiency design by optimal component placement within a rack or space.
Performance Consideration*	• Latency between L1 HWA and baseband SW on CPU • Bandwidth between L1 HWA and baseband SW on CPU • Data integrity measures • Live migration
Management & Orchestration Consideration	The VBS VNF supports several operating configuration to support different radio configurations (supported antenna configuration (2x2/4x4/, etc.) or service requirements such as number of users. Operating configurations may have different CPU, HWA and networking requirements. Hence, resources are assigned for specific base station configuration. The Orchestrator and VIM need to know about available compute resources, acceleration resources and availability of networking routes and features to support: • The VNF (vBS) supported configuration/s CPU resources; • The VNF (vBS) supported configuration/s HWA resources; • The VNF (vBS) supported configuration/s networking resources; – Latency and bandwidth requirements between HWA and CPUs. The VNF Manager and or VNF EMS need to maintain performance, functionality and lifecycle of the VNF VNFCs (CPUs, HWA) and networking resources.
Possible Accelerators	Baseband modem processing unit L1 hardware acceleration might be in the form of: • FPGA • Structured ASIC • Custom ASIC There could be different options for the system implementation of L1 hardware accelerator, as an example: Option 1: • Accelerator Type: In-Line • Accelerator Location: Network Attached Option 2: • Accelerator Type: Look Aside accelerator • Accelerator Location: Bus Attached

Description	The available CPU, Hardware Accelerators and Network nodes resources are identified and reported to the management system. The VBS VNF provider provides descriptors to the orchestrator with possible configurations and related resources per each configuration as well as portability measures (Supported abstraction layers, resources type and features). The Management System need to allocate the resources for the VNF in its specific configuration. Based on VNF EM configuration, the orchestrator and/or VNF EM leverages the VNF related virtualised resources to change and/or optimizes the VNF configuration and performance. The VNF EM and VNF Manager handle performance, functionality and lifecycle of the VNF and its components.
Notes	
Legend: * identify mandatory fields.	

5.1.4 Virtual Acceleration Interface for VNFs

Title	Virtual Acceleration Interface for VNFs (In Server Acceleration)
NFV Components	NFV Infrastructure (NFVI), Management and Orchestration (MANO), Software Architecture, Performance and Security.
Introduction & Problem Statement	With NFVI, Virtual Network Functions (VNFs) run as software-only entities in a hardware agnostic fashion. Examples of VNF range from: • Switching, Routing • CDNs • Security application such as Firewall, Intrusion Prevention systems, Virus and SPAM Protection Systems, IPsec and SSL-VPN gateways • eNodeB • EPC SGW, PGW While a range of VNFs work efficiently as software-only entities, VNFs such as Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), Web Application Firewalls (WAF) that do virus scanning and spam protection, IPsec/SSL-VPN Gateways, LTE requiring Packet Data Convergence Protocol (PDCP) processing and VoIP (Voice over IP) Gateways do compute intensive operations that takes away cycles off the VMs and the VNFs. Achieving high performance for the above mentioned collective umbrella of Compute Intensive applications (CI) is a known challenge when run as VNFs. The CI applications that run on propriety complex hardware-based appliances in a traditional setup (not on cloud or data centres) showcase higher performance as the compute intensive operations (e.g. cryptography, compression/decompression, pattern matching) are offloaded to the hardware accelerators of SoCs. The major stumbling block in providing hardware acceleration for these CIs as VNFs is that the hardware accelerators available today have proprietary vendor specific interfaces that defeat the basic goal of NFV that envisages VNFs to be run as a software-only entity in a hardware agnostic fashion. Keeping the requirement of VNF to achieve high performance virtualised network appliances which are portable between different hardware vendors, it becomes imperative to define a standard vendor independent accelerator interface, Virtual Accelerator Interface, so that VNFs continue to exist as software-only entities and work in a hardware agnostic fashion and yet address the performance challenges for the CI applications as VNFs. In summary, the problem statement is as follows: • CI VNFs are unable to showcase high performances as traditional CIs as they run as software-only entities. Using accelerators is one method with which CI VNFs can showcase higher performance as their traditional counter-parts. • CI VNFs are unable to make use of hardware accelerators as they have proprietary vendor-specific interfaces and using such proprietary interfaces defeats the portability and migration requirements of VNFs across various ecosystems.

Description

Different CI VNFs require specific type of offload accelerators. The table below cites some examples of CI VNFs and the accelerators that they will need.

	VNF Application	Offload Accelerator Capabilities
1	IPsec/SSL Gateway	Symmetric Key Cryptography, Public Key Cryptography, IPsec Protocol Accelerators, SSL Record Layer Accelerators
2	Intrusion Prevention Systems	Pattern matching, Compression, Decompression
3	Web Application, Firewall, Anti-Virus, Anti-Spam Systems	Compression, Decompression, Pattern Matching, SSL Record Layer Processing, Public and Symmetric Cryptography
4	Packet Data Convergence Protocol	Crypto engines Protocol Acceleration
5	VOIP Gateway	Crypto engines SRTP Protocol Acceleration
6	Routing, Firewall	Table lookup Accelerators

The NFV Architectural framework can include a Virtual Accelerator in addition to Virtual Compute, Virtual Storage and Virtual Network that can abstract the underlying proprietary vendor-specific hardware offload accelerators and provide a virtual instance that the VNFs can use. The Virtual Accelerator provide a standardized vendor agnostic accelerator interface that VNFs can use to access the underlying accelerator. The VNFs interface with the Virtual Accelerator using the Virtual Accelerator (VA) Driver.

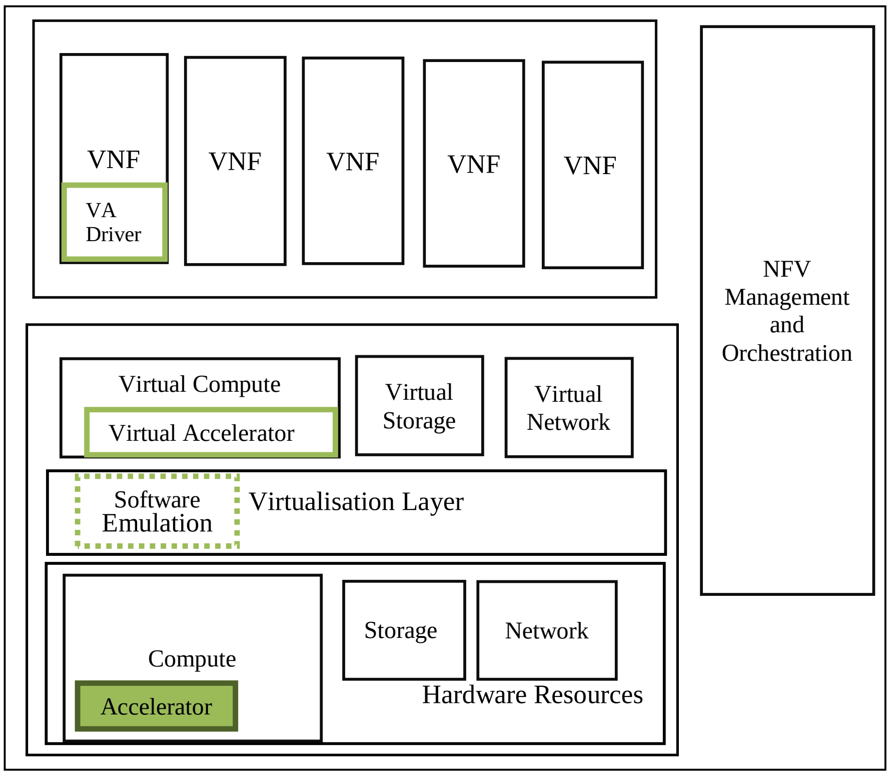
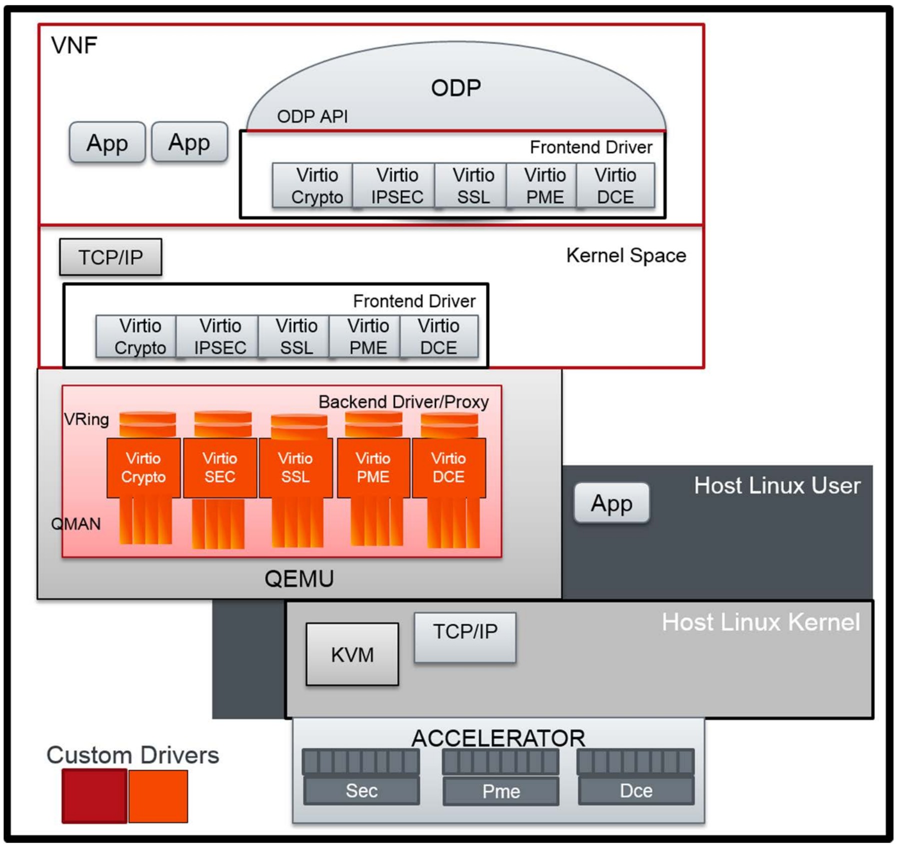
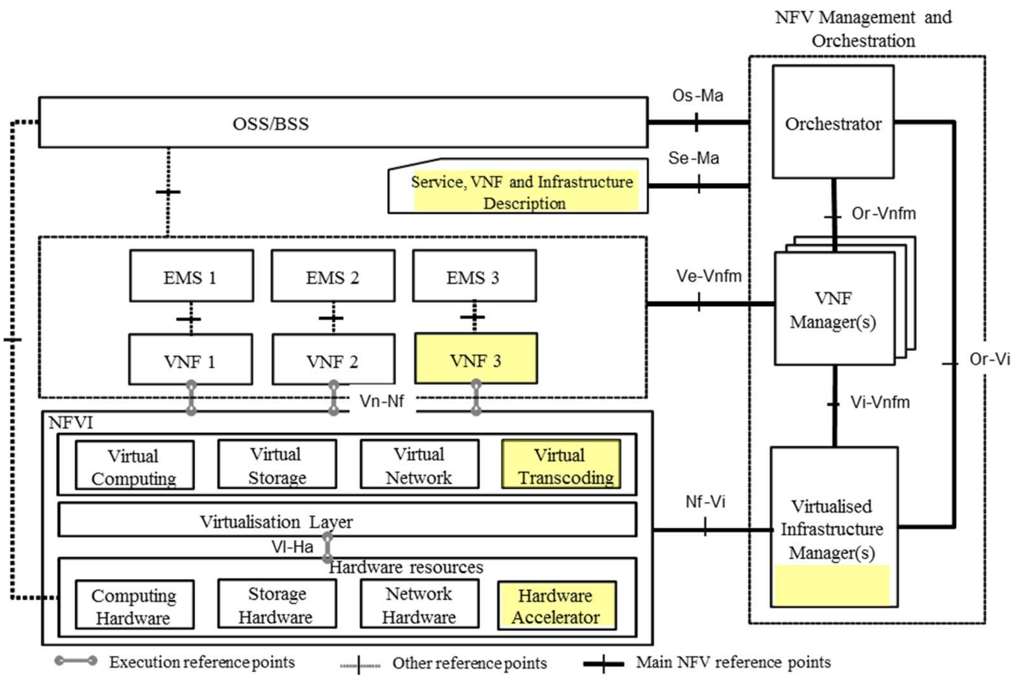


Figure 6: Modified NFV Architectural framework

VA Drivers extend through the range of accelerators such VA-SSL driver, VA-IPsec driver, VA-Pattern Matching driver, etc.
To ensure portability across ecosystems with or without accelerators, the Virtualisation Layer provide a software emulation of the offload functions so that VNFs can continue to exist as software-only entities and work in a hardware agnostic fashion.

	 VNF ODP ODP API App App Frontend Driver Virtio Crypto Virtio IPSEC Virtio SSL Virtio PME Virtio DCE TCP/IP Kernel Space Frontend Driver Virtio Crypto Virtio IPSEC Virtio SSL Virtio PME Virtio DCE Backend Driver/Proxy VRing Virtio Crypto Virtio SEC Virtio SSL Virtio PME Virtio DCE QMAN QEMU App Host Linux User KVM TCP/IP Host Linux Kernel Custom Drivers ACCELERATOR Sec Pme Dce **Figure 7: Standardized Accelerator Interface using Virt-IO drivers** Figure 7 shows a suggested implementation of standardized accelerator interfaces available to VNFs using Virt-IO drivers.
Solution Considerations	The key requirements that need to be met are as follows: • Identify the list of accelerator required to accelerate Compute intensive VNFs. • Define a standard vendor independent accelerator interface for each of the accelerators that can be accessed by VNFs. • Virtual Accelerator works with different proprietary hardware offload accelerators, so that VNFs using this interface can continue to run in a hardware agnostic manner. • Virtual Accelerator interface supports software emulation, so that the VNFs can work seamlessly with or without acceleration. • NFVI supports virtual accelerators in the same fashion as it supports virtual network, storage and compute. • Ability for VNFs to indicate the type of accelerators it could use. • Ability for NFVI to indicate the accelerators it supports.
Performance Consideration	• NFV Accelerator Offload using hardware accelerators can reduce significant compute cycles utilization by the VNFs leaving more for other tasks of VNF and hence result in a capacity gain, throughput gain, connection rate gain or some combination of the above for the VNF. • Avoiding Virtualisation Layer interaction on a per packet basis would bring significant performance gain and hence it is important to consider methods to bypass the Virtualisation layer on a per packet basis.
Management & Orchestration Consideration	MANO match VNF's accelerator's requirements with NFVI's accelerator capabilities.
Possible Accelerators	Crypto - Public key and Symmetric Key, IPsec Protocol Accelerator, SSL Record Layer Accelerators, Pattern Matching, Compression, De-compression, PDCP Accelerator, SRTP Protocol Accelerator and Table Lookup Accelerators Accelerator Type - Look Aside Accelerator Housing/Location - Integrated CPU, iNIC, Bus Attached
Other Consideration	Live Migration Consideration: • When a VNF moves from one NFVI node to another, the VNF continues to work with minimal disruption.

5.1.5 Transcoding

Title	Transcoding Hardware Acceleration Use Case
NFV Components	VNF / VNFC VIM NFVO VNFM
Introduction & Problem Statement *	In the future the rapid adoption of VoLTE, VoWiFi and WebRTC solutions for multimedia communication services will determine the contemporary usage in the network of many different audio and video codecs. Interoperability will require transcoding and/or transrating of the media streams exchanged between end user terminals to adapt them to the capabilities supported by each device. Multimedia services may have high bandwidth requirements, with strict requirements for latency, jitter, and packet loss in order to ensure QoE for end users. Besides, new audio and video codecs such as H.265 or VP9 video require more processing resources than previously used audio codecs. Transcoding is not always performed in "real-time". Anyway the impact on "real-time communication" is more relevant considering the possible impact on the QoE for end-users. Transcoding is performed by decoding a media stream using a specific codec and then re-encoding the information by using a different codec. Transrating is performed by reducing the amount of image data and resolution in order to adapt the video to different available screen sizes and network bandwidth. Media transcoding performed with Hardware accelerators can provide an efficient and effective solution with respect to Software-based transcoding in a fully virtualised environment. Usage of the Hardware acceleration for transcoding needs to be based on the following possible requirements: It can be possible to include Hardware Acceleration capabilities on selected compute servers. It can be possible to create instance of VNFCs making use of Transcoding with virtualisation containers hosted on these servers. It can be possible to make use of VNFC software implementations that are independent from the acceleration technology. It can be possible to include Hardware Accelerators as elements of the NFVI managed by the VIM. It can be possible to perform lifecycle management according to the NFV requirements for the VNF making use of Hardware Acceleration. The impact on the NFV reference architecture is shown in figure 7.  NFV Management and Orchestration OSS/BSS Os-Ma Orchestrator Se-Ma Service, VNF and Infrastructure Description Or-Vnfm EMS 1 EMS 2 EMS 3 Ve-Vnfm VNF Manager(s) VNF 1 VNF 2 VNF 3 Or-Vi Vn-Nf Vi-Vnfm NFVI Virtual Computing Virtual Storage Virtual Network Virtual Transcoding Nf-Vi Virtualisation Layer Virtualised Infrastructure Manager(s) Vl-Ha Hardware resources Computing Hardware Storage Hardware Network Hardware Hardware Accelerator Execution reference points Other reference points Main NFV reference points **Figure 8**

	The major impacts are on the following functions: NFVI can include a new type of Hardware resources providing Transcoding Hardware Acceleration. These resources, shown in figure 8 as "Hardware Accelerator" are part of the Infrastructure "Compute Domain". The virtualisation layer in NFVI can provide "virtual transcoding" resources to the VNFs. VNFD can include the requirements in term of Hardware Acceleration for the VNF and specifically for the VNFCs. VNF can make use of Transcoding Hardware acceleration capabilities (e.g. the VNF implements the Session Border Controller function). Infrastructure Description: compute resources with acceleration capabilities can be identified within the Infrastructure "Compute Domain". The VNFM can be able to request resource allocation according the need of the VNF described in the VNFD. The VIM can be able to select Compute resources with acceleration capabilities.
Performance Consideration *	Performances will be measured with respect to: type and number of transcoded media streams; type of transcoding (e.g. codecs, video resolutions, frames per second); total managed bandwidth; introduced latency (when transcoding real time communication sessions); effect on jitter and packet loss (when transcoding real time communication sessions); energy efficiency (expected lower power consumption).
Management & Orchestration Consideration	Starting from the VNFD and on the knowledge of the resources available on the infrastructure (information regarding consumable virtualised resources that can be provided by the VIM) the VNFM and NFVO may request the allocation of transcoding resources that can fulfil the Hardware acceleration requirements for the VNF. The VIM may support Hardware resources capable of Transcoding Hardware Acceleration. From the VIM N/B interface it may be possible to instantiate, scale, migrate and terminate virtual resources making use of Transcoding Hardware Acceleration.
Possible Accelerators	Possible transcoding hardware accelerators can be implemented with: • PCIe plug-in cards using Digital Signal Processors (DSPs); • SoC; • GPUs; • Extensions of the ISA. Classifying accelerators according to the "type of accelerator" leads to: • Look-aside; • "In-line", • "Fast Path" or "Optimized SW Path". Classifying accelerators according to possible "Housing/Location" classification leads to: • CPU Instruction based, being part of processor instruction set; • Integrated CPU, being housed as a HW function (GPU, FPGA, etc.); • Part of iNIC; • "Network Attached"; • "Bus attached".
Description	Transcoding Hardware accelerator is in the form of specific hardware or chipset. The hardware resource is virtualised with device drivers by the virtualisation layer. NFV Software making use of the Transcoding Hardware Accelerators is the VNF. Specifically, the VNFC instances can perform transcoding by taking advantage of virtualised transcoding resources. The usage of Transcoding Hardware accelerator allows obtaining better performance at lower cost and with lower power consumption.
Legend: * identify mandatory fields.	

5.1.6 Deep Packet Inspection

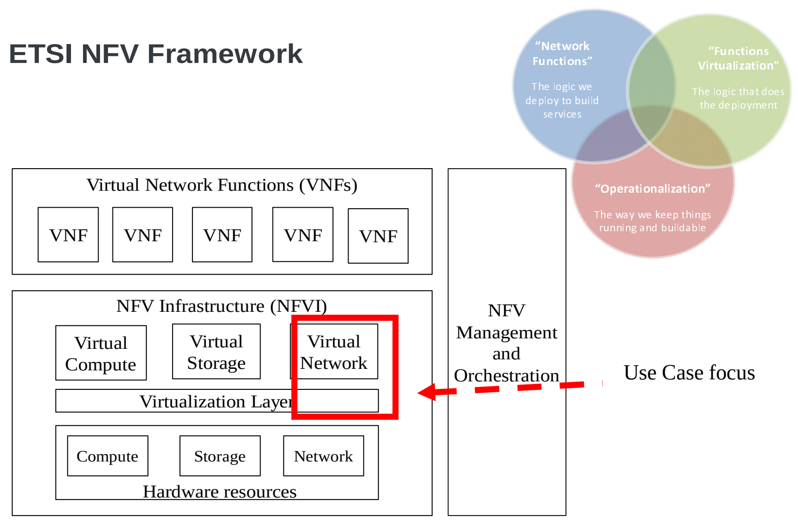
Title	DPI Acceleration Use Case
NFV Components	VNF / VNFC VIM NFVO VNFM
Introduction & Problem Statement *	With the development of mobile internet and 4G, services will become more diverse with different characteristics, like protocol, content type, user-user, user-network and so on. Mobile operators network (i.e. EPC) has the requirement and capability to detect and identify the service data flow, and then they can provide policy control, service data charging, and new business model, like providing sponsored data to users. Deep Packet Inspection (DPI) is usually used in the detection and identification process, especially identifying L4-L7 characteristics of packet or flow. DPI can be performed with accelerators, which can provide a more efficient and effective solution compared to standard COTS implementation in NFV environment. Usage of DPI acceleration function can be based on the following possible requirements: • NFVI can be enhanced to support DPI acceleration capability. • DPI acceleration resources can be managed by the VIM and provided to VNF or VNFC; • DPI acceleration resources can be identified by a VNFD-related information element. Besides, this DPI acceleration function can be used by next generation fire wall.
Performance Consideration *	Performances can be measured with respect to: • Capability on flexible upgrade of match rules. • Number of match rules or number of identified application (protocols, type, and so on). • Total managed bandwidth. • Number of concurrent processing flow. • Introduced latency. • Energy efficiency (expected lower power consumption).
Management & Orchestration Consideration	The major impacts are on the following functions: • NFVI can be enhanced to support DPI acceleration function; • The virtualisation layer in NFVI can provide virtual DPI acceleration resources to the VNFs. • VNFD can include the requirement and related information element, in term of DPI acceleration for the VNF and specifically for the VNFCs. • The VNFM and NFVO can request resource allocation according to the need of the VNF described in the VNFD. • The VIM can manage DPI acceleration capabilities. • VNF(like PGW (Packet Data Network Gateway), NGFW (Next Generation FW)) can make use of DPI acceleration capabilities.
Possible Accelerators	Possible DPI hardware accelerators implementation includes but not limited to: • FPGA • ASIC • Integrated CPU Type of accelerator can be Look-Aside, Fast-Path, In-Line, or Optimized Software Path. Location of accelerators can be: • Integrated CPU • Network Attached • Bus Attached • iNIC
Description	DPI acceleration capability can be a kind of specific hardware or chipset, or software or hybrid implementation. The DPI acceleration resources can be managed by VIM, and described by VNFD-related information elements. The VNFM and NFVO can request resource allocation according to the need of the VNF described in the VNFD. VNF can make use of DPI acceleration capability, i.e. VNFC instances can perform DPI function by taking advantage of virtualised accelerators resources. The usage of DPI acceleration capability should make better performance at acceptable cost and power consumption.
Notes	DPI is one key function of NGFW platform in clause 5.1.2, this use case depicts DPI acceleration in mobile network.
Legend: * identify mandatory fields.	

5.2 Network Acceleration

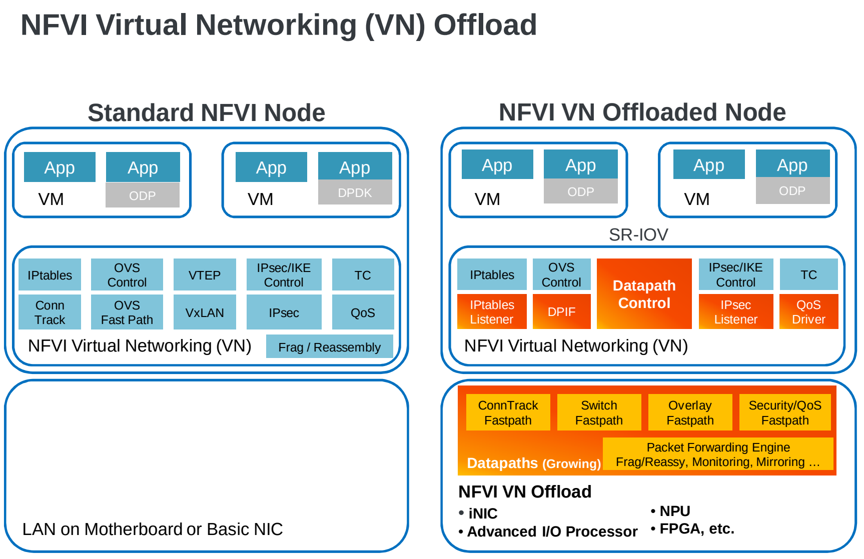
5.2.1 Load Balancing and NAT

Title	Load balancing & NAT
NFV Components	Compute Node, including an intelligent NIC with offload capability.
Introduction & Problem Statement	Load balancing (LB) and NAT have been deployed as a software only solution in the server, such as in the hypervisor, in the form of a vSwitch, or virtual switch. As server network speeds move to 40 GbE it is envisioned that these functions will have to be offloaded, such as to an intelligent NIC in the server, or compute node. This use case can be applied to data centers that use OVS® (Open Virtual Switch).
Performance Consideration	Many applications require the support of multiple physical ports at 40 GbE each. In addition, many compute node servers can host hundreds of VMs, which in turn host many applications. The accelerated solution helps to meet the performance, latency, jitter and the SLA requirements.
Management & Orchestration Consideration	Managing the vSwitch through a local API is required. This acceleration is transparent to the VNFs. It is desirable that the orchestrator is able to discover the underlying performance capability of the compute node.
Possible Accelerators	NFVI Fast Path, In-Line, and Look-Aside Accelerators - all as iNIC, Bus Attached, and Integrated CPU - can all be used for acceleration in this use case.
Description	This use case is comprised of two steps: 1) Load Balance (select a Destination IP address, or DIP). 2) NAT (translate a Virtual IP address, or VIP to a DIP and ports). The virtual switch (e.g. OVS) and the NIC are relevant for the following: • 2nd Tier: Provides connection-level (layer-4) load spreading; implemented in servers; • 3rd Tier: Provides Stateful NAT; implemented in the virtual switch in every server. L4 load spreading among a set of available servers (virtual machines) is implemented by computing a hash function on the traffic received on a given input endpoint. It uses the following fields from an incoming packet to compute a hash value: source and destination IP address, IP protocol (TCP or UDP), source and destination ports. This function would be offloaded to an intelligent NIC in the 2nd Tier.
Other Considerations	One test scenario to benchmark this type of acceleration is as follows: A number of tenants need to be defined in terms of VIPs. VMs are allocated to each tenant and DIPs assigned to them. Multiple VMs are installed in a single server, with OVS and the NIC serving multiple tenants so as to load up the server and CPU cycles adequately and stress the I/O performance in the server. Suggested Benchmarks: 1) LB and NAT are implemented in software and in two tiers; 2) LB is implemented in OVS and NAT is implemented in software, in two tiers; 3) LB is implemented in the NIC and NAT in implemented in software, in two tiers; 4) LB and NAT are implemented in OVS in a single tier; 5) LB and NAT are implemented in the NIC in a single tier.

5.2.2 NFVI Virtual Networking Offload

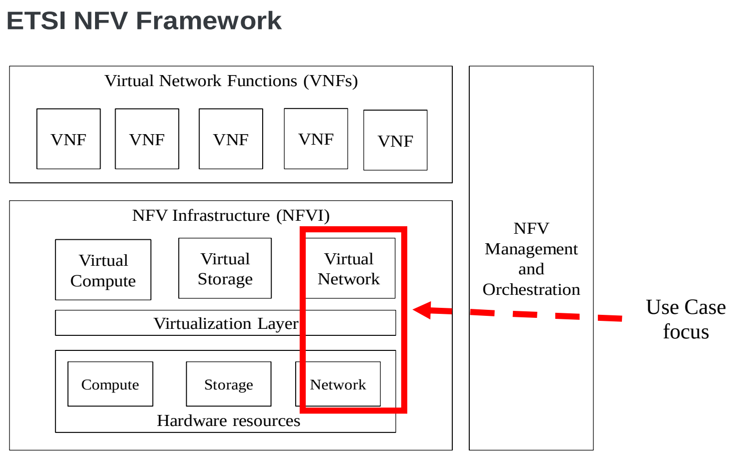
Title	NFVI Virtual Networking Offload
NFV Components	VNF, VIM, Compute Node, Network Node, Storage Node, HWA (Hardware Acceleration) Node.
Introduction & Problem Statement	The key value enabler of NFV is the introduction of virtualisation support in the NFVI that enables service delivery utilizing VNFs decoupled from the underlying physical compute, storage and network infrastructure. By the same token, a key challenge occurs in overcoming the performance impediments that result from the introduction of this virtualisation support. A less obvious but equally important challenge is that these performance impediments need to be overcome in a way that is transparent to VNFs. From figure 9, there are three main components to NFVI virtualisation support: Virtual Compute, Virtual Storage and Virtual Network.  ETSI NFV Framework "Network Functions" The logic we deploy to build services "Functions Virtualization" The logic that does the deployment "Operationalization" The way we keep things running and buildable Virtual Network Functions (VNFs) VNF VNF VNF VNF VNF NFV Infrastructure (NFVI) Virtual Compute Virtual Storage Virtual Network Virtualization Layer Compute Storage Network Hardware resources NFV Management and Orchestration Use Case focus **Figure 9** NFVI Virtual Networking, unlike Virtual Computing and Virtual Storage, can be very CPU intensive, disproportionately bogging down CPU resources available to Virtual Compute/Storage workloads which form the more visible part of the service offering. Furthermore, NFVI Virtual Networking is becoming increasingly burdensome, growing from basic virtual switching to overlay networking to secure transport, QoS, VM isolation/filtering/firewalling and load balancing, etc., not to mention adjunct and utility functions like traffic monitoring, traffic mirroring and fragmentation and reassembly. Therefore, the focus of this Use Case is to describe an approach and high level requirements for overcoming the performance impediments introduced specifically by the networking aspects of the NFVI virtualisation support referenced above as NFVI Virtual Networking in a manner transparent to VNFs and that goes beyond the current state of the art.
Description	NFVI Virtual Networking can be offloaded to mitigate the performance impediment introduced by virtualisation support. One of the key components in NFVI Virtual Networking is the virtual switch. OVS is a popular OpenFlow based software switch implementation that comes with Linux and KVM/QEMU distributions. Therefore, many vendors have focused on OVS acceleration/offload.

	OVS Acceleration/Offload OVS has two components, a user space component and a kernel space component. The user space component consists of OVSDB (configuration), an OpenFlow agent and the OpenFlow normal path processing logic. The kernel component implements the fast path which OVS refers to as the data path. This kernel data path, as is generally the case, is not efficient due to the following overheads: • Linux kernel processing overhead before a packet is handed over to the OVS data path – Interrupt processing – Softirq processing – Dev Layer processing • Interrupt overhead • Fast Path to Normal Path overhead for the first few packets of a flow Some software/networking vendors have accelerated OVS by implementing a user space data path using the OVS "dpif" provider interface and Intel DPDK® or ODP® in poll mode. Poll mode dedicates a few cores to receive packets from the Ethernet controller directly, eliminating interrupts and thereby avoiding overhead associated with interrupt processing. Also, since the Ethernet ports are owned by this custom user space process, there is no need for any complex hook processing as needed in the Linux kernel. This user space data path can start OpenFlow processing on the packets immediately upon reception from Ethernet controllers, thereby avoiding any intermediate processing overhead. These features by themselves provide a good performance boost. Vendors have also offloaded the OVS data path on to an iNIC or NPU using SR-IOV to bypass the virtualisation layer and almost eliminate the OVS data path burden on the general purpose processing layer. **Limitations of OVS Acceleration/Offload** OVS acceleration/offload implementations may not perform well in popular OpenStack environments because these environments depend upon more than just virtual switching. OpenStack environments additionally require the following NFVI Virtual Networking support: • Network virtualisation using VxLAN, NVGRE, etc. • Linux IPtables for isolation amongst VMs • VxLAN-over-IPSec to protect the traffic from eaves dropping • VM connection to OVS via tuntap interfaces Overlay networking, isolation using IPtables firewall, VxLAN-over-IPsec and VM connection via tuntap interfaces all use Linux kernel capabilities. For accelerated OVS data path in user space, packets still have to traverse through the kernel (sometimes twice) to avail of these capabilities beyond virtual switching. This may result in even worse performance than the OVS data path in the kernel. For offloaded OVS, packets will have to traverse between execution domains (i.e. accelerator and general purpose processing layer), potentially multiple times, to avail of these capabilities. Therefore, in an Openstack environment, just OVS acceleration/offload may actually reduce performance for realistic use cases. **NFVI Virtual Network Offload** NFVI Virtual Networking Offload not only envisions offloading of the OVS kernel data path, but also all/most other NFVI virtual networking data/fast paths. By doing so, it enables VNFs to bypass NFVI virtual networking in the general purpose processing layer, as it happens with SR-IOV for example, and thus mitigate performance impediments introduced by virtualisation support without losing functionality provided by this support. Only exception packets go to NFVI Virtual Networking in the general purpose processing layer. The growing list of NFVI Virtual Networking data/fast path components to be offloaded include: • Virtual Switching, e.g. OVS kernel data path • Overlay Networking, e.g. VxLAN, NVGRE, GENEVE • Secure Transport, e.g. IPsec • QoS • VM isolation/filtering, e.g. IPtables, D/DOS, syn flood prevention • Distributed Routing, Firewall and Load Balancer • Traffic Monitoring, e.g. Netflow/Sflow • Traffic Mirroring • Fragmentation/Reassembly • etc.

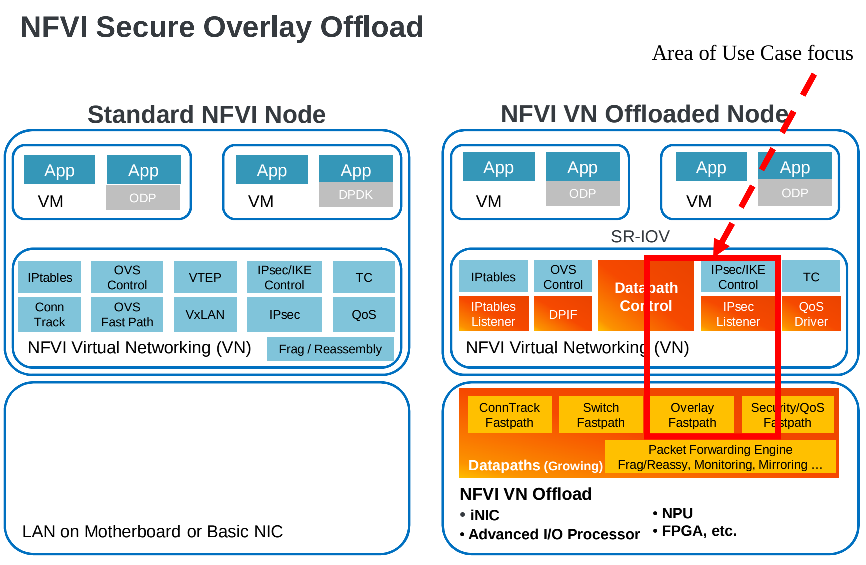
	NFVI Virtual Networking Offload functionality include offloading its various data/fast paths onto the following: • iNICs • Programmable advanced I/O processors • NPUs • FPGAs • etc. Figure 10 shows its various components and organization as compared to the non-accelerated/non-offloaded case.  NFVI Virtual Networking (VN) Offload Standard NFVI Node NFVI VN Offloaded Node App VM ODP DPDK SR-IOV IPtables OVS Control VTEP IPsec/IKE Control TC Conn Track OVS Fast Path VxLAN IPsec QoS NFVI Virtual Networking (VN) Frag / Reassembly Datapath Control IPtables Listener DPIF IPsec Listener QoS Driver LAN on Motherboard or Basic NIC ConnTrack Fastpath Switch Fastpath Overlay Fastpath Security/QoS Fastpath Packet Forwarding Engine Frag/Reassy, Monitoring, Mirroring ... Datapaths (Growing) NFVI VN Offload • iNIC • Advanced I/O Processor • NPU • FPGA, etc. **Figure 10** It is important that NFVI Virtual Networking Offload enable access to offload accelerators, whether it is an iNIC, an advance I/O processor, an NPU, an FPGA, etc. without requiring changes to VNFs.
Performance Considerations	NFVI/VMM Virtual Networking Offload using iNICs, advance I/O processors and/or other forms of HW acceleration needs to reduce CPU utilization leaving more for VNF capacity gains for a given performance profile or increase VNF performance for a given capacity or some combination thereof. This performance improvement implies offloading necessary NFVI virtual networking components as described above. This translates into avoiding intervention by NFVI virtual networking in the general purpose processing layer for processing packets.
Management & Orchestration Considerations	MANO related aspects include: • Ability for NFVI nodes to advertise NFVI Virtual Networking Offload support in terms of performance characteristics such as latency, jitter, throughput and connection rate. • A mechanism for VNFs to request or provide guidance on performance characteristics' constraints/capabilities. • A mechanism to bind VNF requests with NFVI node capabilities.
Possible Accelerators	NFVI Virtual Networking Offload can be realized with iNICs, advanced I/O processors, and/or accelerators leveraging a myriad of accelerator types (e.g. Multi-core Processor, NPU, FPGA, GPU, etc.). There could be different options for the system implementation of NFVI Virtual Networking Offload, e.g.: • NFV Software: – NFVI • Type of Accelerator Type: – Fast Path • Housing/Location of Accelerator: – Integrated CPU – iNIC • Accelerator based on Functional Type: – OF based packet processor

Other Considerations	NFVI Virtual Networking Offload needs to be transparent to VNFs. This also implies that VNFs should support live migration, with or without HW acceleration, and independent of accelerator vendor or type. This is critical because NFV market potential will be primarily dominated by VNF demand and offerings and as a result Network Operators and VNF providers will expect VNFs to be deployable on the NFVI in a HW vendor agnostic manner, with or without HW acceleration, (i.e. standard platforms). Any requirement for vendor specific drivers or software in the VNF to take advantage of the benefits of the vendor specific NFVI HW acceleration will likely be a competitive disadvantage to vendor market share as well as hinder NFV market potential. Even if vendor specific VNF drivers are upstreamed, the resultant driver sprawl could burden VNF validation, maintenance and general life cycle management. NFVI Virtual Networking Offload needs to also be easily upgradeable and programmable to meet the increasing and changing NFVI virtual networking needs.
Solution Considerations	• NFVI/VMM Virtual Networking Offload using iNICs, advance I/O processors and/or other forms of HW acceleration needs to reduce CPU utilization leaving more for VNF capacity gains for a given performance profile or increase VNF performance for a given capacity or some combination thereof. – Minimize NFVI Virtual Networking in the general processing layer by offloading it to an offload accelerator directly accessible to VNFs (e.g. SR-IOV). – Only exception processing is handled by NFVI virtual networking in the general processing layer. • NFVI Virtual Networking Offload needs to be transparent to VNFs. – VNFs do not need any NFVI accelerator specific software, even if the virtualisation layer is bypassed (e.g. SR-IOV). For example, no special Ethernet drivers is needed in the VNF. Hence, a standardized Ethernet interface is needed across vendors. – VNFs supporting live migration, with or without HW acceleration, and independent of accelerator vendor or type. • NFVI Virtual Networking Offload needs to also be easily upgradeable and programmable to meet the increasing and changing NFVI virtual networking needs.

5.2.3 NFVI Secure Overlay Offload

Title	NFVI Secure Overlay Offload
NFV Components	VNF, VIM, Compute Node, Network Node, Storage Node, HWA (Hardware Acceleration) Node.
Introduction & Problem Statement	The key value enabler of NFV is the introduction of virtualisation support in the NFVI that enables service delivery utilizing VNFs decoupled from the underlying physical compute, storage and network infrastructure. By the same token, a key challenge occurs in overcoming the performance impediments that result from the introduction of this virtualisation support. A less obvious but equally important challenge is that these performance impediments need to be overcome in a way that is transparent to VNFs. From the figure 11 below, there are three main components to NFVI: Virtual Compute, Virtual Storage and Virtual Network.  ETSI NFV Framework Virtual Network Functions (VNFs) VNF VNF VNF VNF VNF NFV Infrastructure (NFVI) Virtual Compute Virtual Storage Virtual Network Virtualization Layer Compute Storage Network Hardware resources NFV Management and Orchestration Use Case focus **Figure 11** NFVI Virtual Networking, unlike Virtual Computing and Virtual Storage, can be very CPU intensive, disproportionately bogging down CPU resources available to Virtual Compute/Storage workloads which form the more visible part of the service offering. Furthermore, NFVI Virtual Networking is becoming increasingly burdensome. The growing list of NFVI Virtual Networking functions include: • Virtual Switching, e.g. OVS kernel data path • Overlay Networking, e.g. VxLAN, NVGRE, GENEVE, Custom Tunneling • Secure Transport, e.g. IPsec • QoS • VM isolation/filtering, e.g. IPtables, D/DOS, syn flood prevention • Distributed Routing, Firewall and Load Balancer • Traffic Monitoring, e.g. Netflow/Sflow • Traffic Mirroring • Fragmentation/Reassembly • etc. Therefore, the focus of this Use Case is on NFVI Virtual Networking. Specifically, this Use Case homes in on the need for the Overlay Networking and Secure Transport (or Secure Overlay in brief) subcomponents of NFVI Virtual Networking and describes an approach and high level requirements for overcoming the performance impediments introduced by these subcomponents in a manner transparent to VNFs.

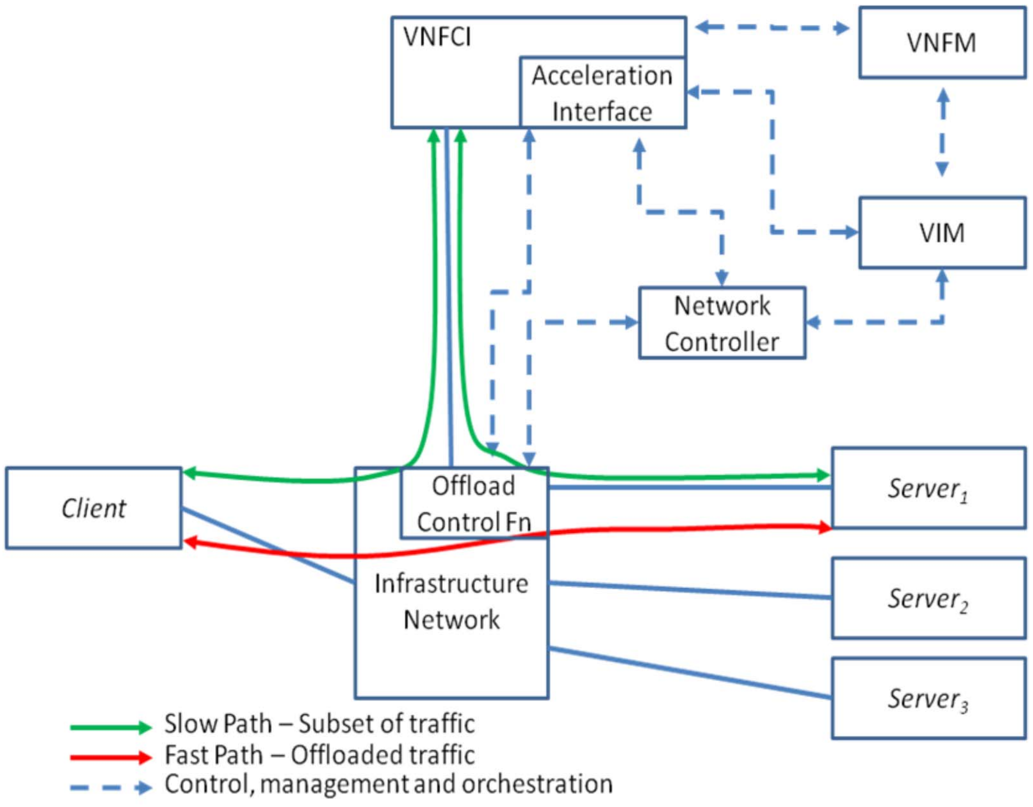
Description	**Overlay Networking** While Virtual Compute and Virtual Storage leverage maturing cloud technologies, the core of Virtual Network is increasingly contemplated as L2inL3 Overlay Networks to overcome VLAN scalability limitations for large and hyper scale cloudified networks envisioned by NFV. • VLANs are fixed in number. The VLAN header defines 12 bits for VLAN ID which means only 4K VLAN IDs are possible. Assuming the best case of 1 VLAN ID per tenant, at most only 4K tenants can be supported. • VLANs are mostly an L2 concept. Keeping VLANs intact across L2 networks separated out by L3 routers is not straightforward and requires some intelligence in L3 devices. Especially when tenant networks need to be expanded to multiple geographic locations, extending VLANs across the Internet requires newer protocols (such as TRILL). • If tenants require VLANs themselves for various reasons, double or triple tagging may be required. Though 802.1ad tagging can be used for tenant identification and 802.1Q tagging for tenant specific VLANs, this may also require changes to existing devices. A popular overlay technology is VxLAN named as such to be indicative of an extension to VLANs, overcoming the limitations of this traditional work horse of virtual networks. VxLAN is a new tunneling protocol that works on top of UDP/IP. It does require changes to existing infrastructure for its support, but it does not have the limitations of VLAN based tenant identification. Since L2 networks are created over L3 networks, they can now easily extend not only within a Data Center/Enterprise/Provider locations, but across different locations of Data Center/Enterprise/Provider networks. VxLAN or other overlay networking protocols are the key enabling technologies to large scale Network Virtualisation and Multi-Tenancy. **Secure Overlay** Security considerations are critical as networks become virtual with overlay technologies like VxLAN. For example, it is possible to corrupt the VTEP (VxLAN Tunnel End Point) learning tables by the man-in-the middle or even external attackers. VNIs can become known to attackers eventually. With this knowledge, multicast or unicast packets can be generated to corrupt or overwhelm learning tables, thereby creating a DoS condition. IPsec amongst VTEPs can mitigate these issues with its support of network-level data integrity, data confidentiality, data origin authentication, and replay protection. The Management Entity can populate IPsec keys in the VTEPs. For Multicast tunnels, the IPsec key needs to be the same across all VTEPs. The Management Entity may recycle the key to strengthen security. For unicast VxLAN tunnels, the Management Entity can either use the same key for all VTEPs or it can use pair wise keys. With proper precautions virtual networks can be just as secure as regular networks.

Description	**Secure Overlay Offload** As part of overall NFVI Virtual Networking Offload, NFVI Secure Overlay Offload is highlighted in figure 12.  NFVI Secure Overlay Offload Area of Use Case focus Standard NFVI Node NFVI VN Offloaded Node App VM ODP DPDK SR-IOV IPtables OVS Control VTEP IPsec/IKE Control TC Conn Track OVS Fast Path VxLAN IPsec QoS NFVI Virtual Networking (VN) Frag / Reassembly Datapath Control IPtables Listener DPIF IPsec Listener QoS Driver LAN on Motherboard or Basic NIC ConnTrack Fastpath Switch Fastpath Overlay Fastpath Security/QoS Fastpath Packet Forwarding Engine Frag/Reassy, Monitoring, Mirroring … Datapaths (Growing) NFVI VN Offload • iNIC • Advanced I/O Processor • NPU • FPGA, etc. **Figure 12** It is important that NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, enable access to offload accelerators, whether it is an iNIC, an advance I/O processor, an NPU, an FPGA, etc., without requiring changes to VNFs.
Performance Considerations	NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, using iNICs, advance I/O processors and/or other forms of HW acceleration needs to reduce CPU utilization leaving more for VNF capacity gains for a given performance profile or increase VNF performance for a given capacity or some combination thereof.
Management & Orchestration Considerations	MANO related aspects include: • Ability for NFVI nodes to advertise support for NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, in terms of performance characteristics such as latency, jitter, throughput and connection rate. • A mechanism for VNFs to request or provide guidance on performance characteristics' constraints/capabilities. • A mechanism to bind VNF requests with NFVI node capabilities.
Possible Accelerators	NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, can be realized with iNICs, advanced I/O processors, and/or accelerators leveraging a myriad of accelerator types (e.g. Multi-core Processor, NPU, FPGA, GPU, etc.) There could be different options for the system implementation of NFVI Secure Overlay Offload, e.g.: • NFV Software: – NFVI. • Type of Accelerator Type: – Fast Path. • Housing/Location of Accelerator: – Integrated CPU; or – iNIC. • Accelerator based on Functional Type: – OF based packet processor.
Other Considerations	NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, needs to be transparent to VNFs. This also implies that VNFs should support live migration, with or without HW acceleration, and independent of accelerator vendor or type. NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, needs to also be easily upgradeable and programmable to meet the increasing and changing NFVI virtual networking needs.

Summary	• NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, using iNICs, advance I/O processors and/or other forms of HW acceleration needs to reduce CPU utilization leaving more for VNF capacity gains for a given performance profile or increase VNF performance for a given capacity or some combination thereof. – Minimize NFVI Virtual Networking in the general processing layer by offloading it to an offload accelerator directly accessible to VNFs (e.g. SR-IOV). – Only exception processing is handled by NFVI virtual networking in the general processing layer. • NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, needs to be transparent to VNFs. – VNFs shall not need any NFVI accelerator specific software, even if the virtualisation layer is bypassed (e.g. SR-IOV). For example, no special Ethernet drivers should be needed in the VNF. Hence, a standardized Ethernet interface is needed across vendors. – VNFs should be live migratable, with or without HW acceleration, and independent of accelerator vendor or type. • NFVI Secure Overlay Offload, as part of overall NFVI Virtual Networking Offload, needs to also be easily upgradeable and programmable to meet the increasing and changing NFVI virtual networking needs.

5.2.4 Dynamic Optimization of Packet Flow Routing

Title	Dynamic Optimization of Packet Flow Routing (DOPFR) (ETSI GS NFV-INF 005 [i.3])
NFV Components	VNF, VIM, Infrastructure Network including Network Controller.
Introduction & Problem Statement	VNFs may take advantage of the capabilities to offload traffic to the infrastructure network. This follows SDN principles of separation of control and data plane, where the control plane may be implemented in the VNFC instance (VNFCI) and the data plane offloaded to the infrastructure network resources. This requires the infrastructure network to be able to manage the state and actions associated with a very large number of flows and high flow modification rate.
Performance Consideration	Taking advantage of the existing capabilities of the infrastructure network improves VNF packet processing performance while decreasing the usage of server resources.
Management & Orchestration Consideration	The VIM requests the Network Controller to provide a logical switch dedicated to the VNF that will be exposed to the VNFCI so it can request DOPFR for some of its flows to dynamically reroute over its internal VNF connectivity without impacting the other Infrastructure Network resources. The VIM also configures a virtual link to enable establishment of a communication channel (e.g. OpenFlow) which will be used by the VNFCI to offload packet processing to the logical switch.
Possible Accelerators	Fast Path Packet Processors. Possible locations include: – Integrated CPU; – iNIC; – Network Attached; – Processor Interconnect Attached. It includes physical switches, virtual switches and other accelerator devices.

Description	 VNFCI Acceleration Interface VNFM VIM Network Controller Client Offload Control Fn Infrastructure Network Server1 Server2 Server3 Slow Path – Subset of traffic Fast Path – Offloaded traffic Control, management and orchestration **Figure 13** VNFCI can request offloads at run-time via the Network Controller or be accelerated with an offload control function that resides directly on the infrastructure network resources that perform the acceleration. For example, a virtual Layer 7 Load Balancer (vL7LB) may have a slow path consisting of the 3 way handshake for the TCP connections that allows the vL7LB to make the load balancing decision. Then, the vL7LB instructs the infrastructure network to perform the proxy phase where the infrastructure network redirects the TCP segments from the client to the server TCP segments and vice versa, i.e. the fast path. This redirection requires traffic be rerouted in the infrastructure network and some actions applied to the packets.
Other Considerations	

5.3 Storage Acceleration

5.3.1 NVMe™ Over Fabric Enabled Acceleration

Title *	NVMe™ Over Fabric Enabled Acceleration
NFV Components *	VNF/VNFC/VNFD VIM Compute Node Storage Node/Virtual Storage
Introduction & Problem Statement *	In order to achieve high performance in storage systems, ephemeral storage and cache acceleration are currently widely adopted. However in NFV, when operators have high QoS requirements and large storage system, these mechanisms might not perform so well. Hence in this use case, NVMe™ SSD is proposed to replace the traditional SSD to speed up cache acceleration. Also, NVMe™ Over Fabric (NOF) technique is proposed here to accelerate remote storage access.
Performance Consideration *	a) Latency between storage server and JBOD. b) Latency between compute and storage server. c) Number of remote storage end-points.
Management & Orchestration Consideration *	**VNFD** Requirements and Reports would be provided via VNFD. For example Orchestrator would inform VIM about certain NVMe™ Over Fabric performance requirement through VNFD. **VIM** VIM would have the ability of maintaining the lifecycle of the acceleration, providing support for VNF to be scheduled with efficient storage resource, and completing storage side of live migration if necessary.
Possible Accelerators	a) Type : In-Line Accelerator b) Location: Network Attached c) Examples: – NVMe™ Controller, which is a chip located in NVMe™ device. – RNIC(RDMA Network Interface Card) is necessary for remote NVMe disks are plugged into the system.
Description *	**VNF / VNFC** By the support of concept like Virtual Functions (VF) and Namespaces, multiple VNFCs or VNFs on a single compute node could be attached to multiple VFs virtualised on NVMe™ Controller side via NVMe™ driver. **Compute / Storage Node / Virtual Storage** Different VFs would interact with corresponding Namespaces on NVMe™ SSD to support storage multi-tenancy. Compute Node would need to rely on RNIC to communicate between local and remote NVMe™ instances.
Other Considerations	Security issue regarding NOF is still under study.
Legend: * identify mandatory fields.	

5.3.2 High Performance Persistent Memory on Compute Node

Title *	High performance persistent memory on compute node
NFV Components *	Compute Node VNF Manager VIM Orchestrator
Introduction & Problem Statement *	Network functions such as DNS servers and HSS require very low latency access to large persistent datasets to perform their duties. The typical decoupling of compute and storage described in NFV may jeopardize performance goals. To cope with ever increasing demands, new technologies such as NV-DIMM and Flash-DIMM are now available to bare metal applications. NV-DIMM is to be used for highest performance, sub terabyte datasets such as DNS servers. Flash-DIMM is more likely to be used for multi-terabyte datasets that require less than hundreds of microsecond latencies such as HSS Databases.
Performance Consideration *	Latency to read or write a record Jitter of read or write operations Number of queries and updates per second Dataset capacity Data integrity measures
Management & Orchestration Consideration *	VIM and Orchestrator need to know about available capacity, location and mode of access of persistent memory. VIM has to maintain usage count.
Possible Accelerators	Accelerator types: Optimized Store Examples: NV-DIMM: regular DDR3 DIMM plus Flash backup powered by hypercapacitors, Flash DIMM: Flash memory chips on a DDR3 DIMM module (no DDR3 memory). Accelerator locations: Memory Slots, Bus Attached, Network Attached VNF leveraging of accelerators shall be independent from accelerator types and locations.
Description *	Depending on the technology used, usage of the accelerator can be as simple as making use of a specified memory zone or a driving PCI device. For NV-DIMM and some Flash DIMM technologies, the available persistent memory is identified by the VIM through BIOS/EFI memory map. It is reserved by the VIM and made available to the VNFC via the hypervisor. Flash DIMM may also present a specific PCI interface which looks similar to NVMe™ device except that the controlling PCI registers are in memory rather than on PCI configuration space.
Other Considerations	
Legend: * identify mandatory fields.	

Annex A (informative): Authors & contributors

The following people have contributed to this specification:

Rapporteur:
Jinwei Xia, Huawei

Other contributors:
Nabil G. Damouny, Netronome

Ning Zong, Huawei

François-Frédéric Ozog, 6WIND

Srini Addepalli, Freescale

Subhashini Venkataraman, Freescale

Bharat Mota, Freescale

Zhipeng Huang, Huawei

Ron Breault, Wind River

Alfred Morton, AT&T

Robert Dimond, ARM

Eran Bello, Asocsnetworks

Yuan Yannan, China Mobile

Andy Reid, BT

Skerry Brian J, Intel

Bruno Chatras, Orange

Giuseppe Monteleone, ITALTEL SpA

Andrew Thurber, Cisco

Evelyne Roch, Huawei

Janusz Pieczerak, Orange

History

Document history		
V1.1.1	December 2015	Publication