ETSI GS NFV-IFA 002 V2.1.1 (2016-03)

Network Functions Virtualisation (NFV); Acceleration Technologies; VNF Interfaces Specification

Disclaimer

The present 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-IFA002

Keywords
acceleration, interoperability, NFV, NFVI,
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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

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 (https://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**", "**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 specifies requirements for a set of abstract interfaces enabling a VNF to leverage acceleration services from the infrastructure, regardless of their implementation. The present document also provides an acceleration architectural model to support its deployment model.

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 https://docbox.etsi.org/Reference.

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

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

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

2.2 Informative references

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

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

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

[i.1] ETSI GS NFV-INF 003: "Network Functions Virtualisation (NFV); Infrastructure; Compute Domain".

[i.2] ETSI GS NFV-SWA 001: "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture".

[i.3] ETSI GS NFV-IFA 003: "Network Functions Virtualisation (NFV); Acceleration Technologies; vSwitch Benchmarking and Acceleration Specification".

[i.4] ETSI GS NFV-IFA 004: "Network Functions Virtualisation (NFV); Acceleration Technologies; Management aspects Specification".

[i.5] NVM Express™ Inc: NVM Express 1.0e, NVM Express 1.1, NVM Express 1.2.

NOTE: Available at http://www.nvmexpress.org/specifications/.

[i.6] ETSI GS NFV-INF 005: "Network Functions Virtualisation (NFV);Infrastructure; Network Domain".

[i.7] ETSI GS NFV-IFA 001: "Network Functions Virtualisation (NFV); Acceleration Technologies; Report on Acceleration Technologies & Use Cases".

3 Definitions and abbreviations

3.1 Definitions

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

abstract interface: computer specification and modelling construct. It defines an information model and a way to communicate between two or more entities

NOTE: Computing objects and APIs can be developed for a programming language to implement it.

Application Programming Interface (API): computer programming language construct, composed of a set of functions, methods, objects, structures and constant definitions used to implement an abstract interface

NOTE: This construct is called an abstract interface language binding. There might be multiple bindings to a single abstract interface, one for each computer language (C, C++, Java™, Ruby, PHP, etc.).

dispatching: deterministic method of distributing packets amongst packet queues to be handled by the network stack, exposed as a NIC capability

NOTE: This dispatching can be done by an operating system or a software library such as DPDK or ODP. The criteria for dispatching can be as simple as round robin or as complex as packet hashing on combination of IP address, TCP ports, taking into account tunnelling. The number of processor threads handling the queues can be lower or higher than the number of those queues.

extensible para-virtualised device: para-virtualised device that can execute code and use resources provided by the hypervisor domain at runtime

NOTE: The benefit of the extensibility is to avoid crossing virtualisation boundary whenever it is possible. The resources enable bypassing the hypervisor in a hardware independent manner.

Execution Environment (EE): set of resources and control mechanisms on which application are running

NOTE 1: Examples of Execution Environments include:

- Traditional Operating System.
- RUMP kernels.
- Java™ Virtual Machine (with or without underlying Operating System).
- Xen Minios.
- DPDK.
- Docker.

NOTE 2: An Execution Environment may be virtualised on top of a hypervisor or not.

NOTE 3: Execution Environments may share or not resources such as processor between applications running on top of them.

language binding: API definition for an abstract interface on a particular programming language

library: collection of implementations of behaviour, written in terms of a language that has a well-defined interface by which the behaviour is invoked

NOTE: This means that as long as a higher level program uses a library to make system calls, it does not need to be re-written to implement those system calls over and over again. In addition, the behaviour is provided for reuse by multiple independent programs. A program invokes the library-provided behaviour via a mechanism of the language. For example, in a simple imperative language such as C, the behaviour in a library is invoked by using C's normal function-call. What distinguishes the call as being to a library, versus being to another function in the same program, is the way that the code is organized in the system.

load balancing: dynamic application level traffic distribution function, that distributes traffic amongst VNFC instances within a VNF or amongst VNF instances

NOTE: As defined in ETSI GS NFV-SWA 001 [i.2], clause 5.1.4.

offload: delegate processing (e.g. classification, forwarding, dispatching, load balancing, cryptography, and transcoding) to a different processor or other specialized hardware entity

Real Time Application (RTA): application whose execution can be **guaranteed** to be accomplished under a specific "execution contract"

NOTE: For instance within a bounded delay, for a certain bandwidth. It assumes execution on a Real Time Execution Environment or Operating System (see Real Time Execution Environment).

Real Time Execution Environment (RTEE): Execution Environment that offer applications with provisions to meet their execution contract (see Real Time Application)

Real Time Operating System (RTOS): Real Time Execution Environment in the form of an Operating System

NOTE: An RTOS has an advanced algorithm for scheduling. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency; a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

software framework: abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software

NOTE 1: A software framework is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions. Software frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution.

NOTE 2: Frameworks contain key distinguishing features that separate them from normal libraries:

- Inversion of control: In a framework, unlike in libraries or normal user applications, the overall program's flow of control is not dictated by the caller, but by the framework.
- Default behaviour: A framework has a default behaviour. This default behaviour is some useful behaviour and not a series of no-ops.
- Extensibility: A framework can be extended by the user usually by selective overriding or specialized by user code to provide specific functionality.
- Non-modifiable framework code: The framework code, in general, is not supposed to be modified, while accepting user-implemented extensions. In other words, users can extend the framework, but should not modify its code.

3.2 Abbreviations

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

API	Application Programming Interface
Encrypt/Decrypt	Encryption and Decryption
Encap/Decap	Encapsulation and Decapsulation
EPD	Extensible Para-virtualised Device
IRQ	Interrupt ReQuest
NAT	Network Address Translation
NUMA	Non Uniform Memory Access
NIC	Network Interface Card
RAM	Random Access Memory
SA	Security Association
SPD	Security Policy Database
UML	Unified Modelling Language

VA	Virtual Accelerator
VNI	VXLAN Network Identifier
VTEP	VXLAN Tunnel End Point
VXLAN	Virtual eXtensible Local Area Network

4 Overview

4.1 Problem Statement

4.1.1 VNF Acceleration goals

The goals of the present document are:

- to identify common design patterns that enable an executable VNFC to leverage, at runtime, accelerators to meet their performance objectives;
- to describe how a VNF Provider might leverage those accelerators in an implementation independent way; and
- to define methods in which all aspects of the VNF (VNFC, VNFD, etc.) could be made independent from accelerator implementations.

VNF providers have to mitigate two goals:

- VNFs might have constraints to perform their function within certain power consumption boundaries, CPU core count, PCI express slot usage and with good price/performance ratio; and
- VNFs should accommodate most if not all deployment possibilities.

Use of accelerators can help meet the constraints but can have an influence on deployment flexibility. VNF acceleration implementations will range from inflexible software that is tightly-coupled to specific software and hardware in the VNF and NFVI (see pass-through model as shown in figure 4.1.1-1), to highly flexible loosely-coupled software that uses common abstractions and interfaces (see abstracted model as shown in figure 4.1.1-2). Further it is understood that virtualisation and acceleration technologies will evolve. Pass-through deployments are expected to exist, and the present document does not intend to preclude any specific acceleration architectures from NFV deployments. However, the focus of the present document is to define and promote abstracted models.

It is desirable that the use of accelerators be implementation independent. There is a slight difference between "implementation independent executable VNFC" and "implementation independent VNF":

- An implementation independent executable VNFC is a software that can leverage a known set of accelerator implementations both in hardware and in software. The part of the VNFD that applies to this VNFC contains information elements that allow the NFV management and orchestration to find a compute node with the requested characteristics or hardware. Should a new hardware become available on the market, a VNF Provider willing to make use of it to accelerate a VNFC has to update it's the VNFC image and the VNFD, the operator has to on-board the new VNF package and redeploy it to make use of the new hardware. This was defined as the pass-through model in ETSI GS NFV-INF 003 [i.1], clause 7.2.2.

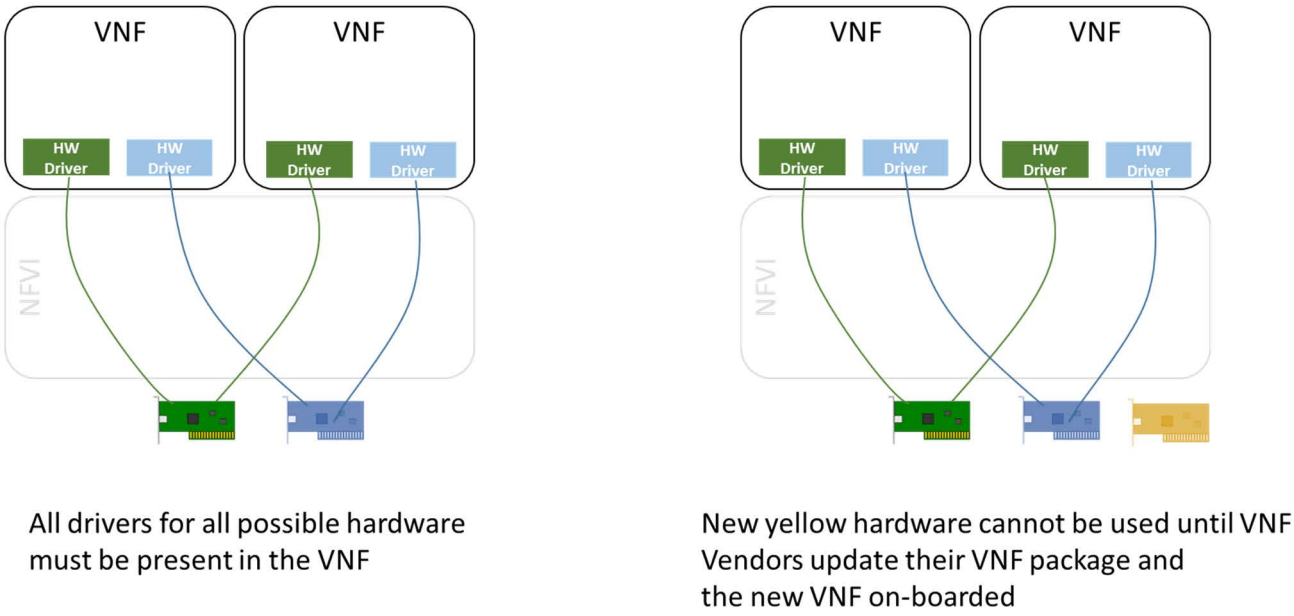

Figure 4.1.1-1: Pass-through model

- An implementation independent VNF is a VNF that makes no assumption whatsoever on the underlying NFVI. Its VNFD does not contain any accelerator specific information elements. Should a new hardware become available on the market, the operator will update its NFVI to allow the VNF to make use of the new hardware. An implementation independent VNF is thus based on implementation independent VNF software that makes use of a functional abstraction of an accelerator supported by an adaptation layer in the NFVI. This model is close to the abstracted model defined in ETSI GS NFV-INF 003 [i.1], clause 7.2.2.

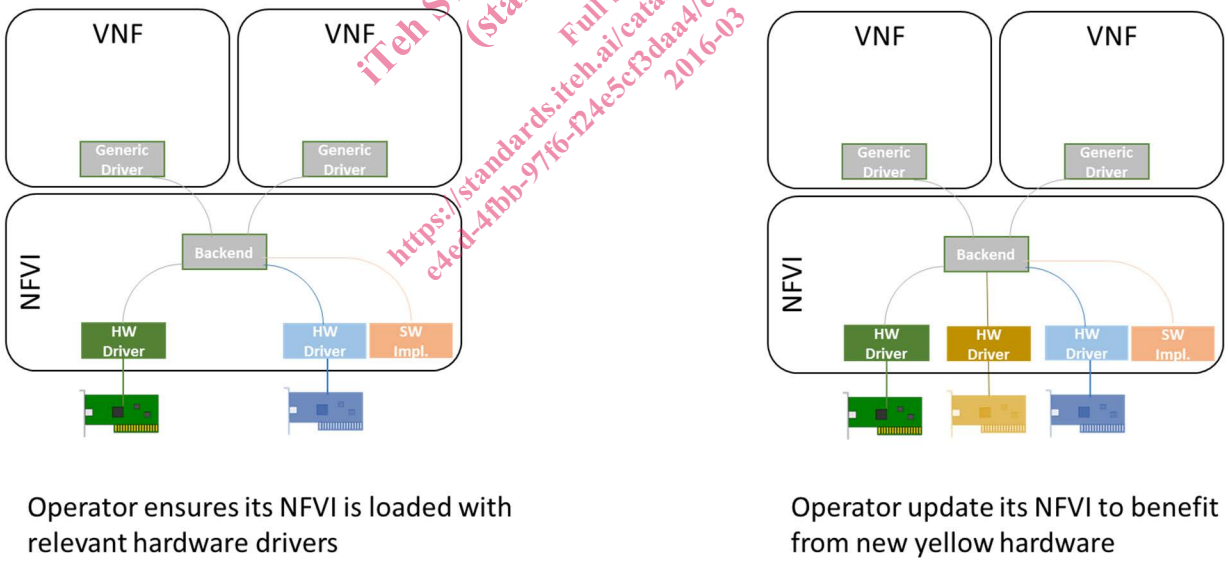

Figure 4.1.1-2: Abstracted model

NOTE 1: An implementation independent executable VNFC allows for VNF deployment in both hypervisor based and non-hypervisor based environments. The latter configuration is outside the scope of the present document.

Live migration of such hardware independent accelerated VNF may be possible if any associated acceleration state information required can also be migrated.

NOTE 2: Live migration from a compute node with accelerator to a compute node without accelerator or with a different accelerator is allowed (in particular to cope with emergency response situations).

A further refinement of the aforementioned goals is to make sure that VNFs can leverage those accelerators regardless of:

- the diversity of VNF software execution environments:
  - Operating Systems (Windows®, Linux®, etc.);
  - Java™ Virtual Machine;
  - RUMP Kernels;
  - DPDK, ODP;
- the diversity of software frameworks or libraries for acceleration, each having provisions to leverage software or hardware acceleration technologies; and
- the diversity of virtualisation environments such as KVM, Xen, VMWare ESX or Microsoft® Hyper-V.

NOTE 3: The present document focuses on acceleration of the VNF execution, it does not cover how a VNF influences packet forwarding in the NFV,which is covered by ETSI GS NFV-IFA 003 [i.3] or by ETSI GS NFV-IFA 004 [i.4]. Requirements for the packet dispatching control interface are however specified by the present document.

NOTE 4: "Microsoft®, Windows®, Linux® and Java™ are examples of a 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".

The accelerators considered span several different execution aspects:

- network stack acceleration: Packet dispatching, manycasting, partial networking stack offloads (L4, IPSec, etc.);
- network payload acceleration: compression, transcoding, pattern matching;
- cryptography (encryption, hashing, random number generation, etc.);
- storage;
- digital Signal Processing; and
- algorithmic Acceleration.

4.1.2 Network related acceleration

The network related accelerations have to take into account the network overlays involved. While many NICs offer TCP accelerations (checksum calculation and verification, TCP Segmentation Offload, etc.) some cannot operate when the traffic is in overlay networks (either layer 2 such as VxLAN or layer 3). Furthermore, when the traffic itself is tunnelled, for instance GTP for a GGSN node, TCP accelerations on such traffic is to be considered as partial networking stack offload.

4.1.3 Storage related acceleration

There is local storage on the compute node and remote storage on other NFVI nodes. The acceleration considers how both types of storage are accessed from the VNF.

4.1.4 Algorithmic acceleration

Algorithmic acceleration can be used by VNFs and NFVI within an NFV environment in order to achieve a better performance and more deterministic behaviour when executing algorithmically complicated functions.