Towards a Flexible Architecture for Virtualized Network Function Platforms

Vinícius Fülber Garcia[1], Leonardo da Cruz Marcuzzo[2], Carlos Raniery de Paula dos Santos[2], Elias Procópio Duarte Junior[1]

[1]Federal University of Paraná, Curitiba, Paraná, Brasil
[2]Federal University of Santa Maria, Santa Maria, Rio Grande do Sul, Brasil
[1]{vfgarcia, elias}@inf.ufpr.br, [2]{lmarcuzzo, csantos}@inf.ufsm.br

07 May 2019

INTRODUCTION

Network Function Virtualization (NFV) is a paradigm which aims to decouple network functions from proprietary hardware and deploy them in a software plane (NFVISG, 2012). Some benefits of NFV paradigm are:

- Fully supported by current virtualization techniques
- Low CAPital and OPerational EXpenditures (CAPEX and OPEX)
- Management and maintenance flexibility
- Fast prototyping and low time-to-market of new network functions

According to the European Telecommunications Standards Institute (NFVISG, 2014), a Virtualized Network Function (VNF) is the implementation of a network function on an NFV infrastructure. Furthermore, a VNF is composed of two main parts (GARCIA et al., 2019):

1. VNF Platform
2. Network Function (NF)

Currently, VNF platforms are released without any standardization of their internal modules and connections, effectively resulting in proprietary solutions. In this context, we present an overview of a generic and flexible VNF platform architecture. Besides foreseeing the adoption of Network Service Header (NSH) (QUINN; ELZUR; PIGNATARO, 2018) and customized monitoring techniques, the architecture also enables the definition and execution of multiple VNF Components (VNFC) (NFVISG, 2014) connected by an internal router. These characteristics facilitate the development of holistic and modular VNF platforms, where the internal modules can be easily swapped in order to create VNFs tailored to specific deployment needs.

VIRTUALIZED NETWORK FUNCTION PLATFORM ARCHITECTURE

The proposed architecture for VNF platforms, depicted in Figure 1, consists of six main modules deployed on a host operating system (here called the VNF Core). Each module performs specific operations within the VNF and can process network packets from both the data and control planes – depicted by solid lines – and the management plane – depicted by dashed lines. External interfaces to the VNF Core are also defined in the architecture to enable the use of virtualized resources and to support both management and orchestration operations.

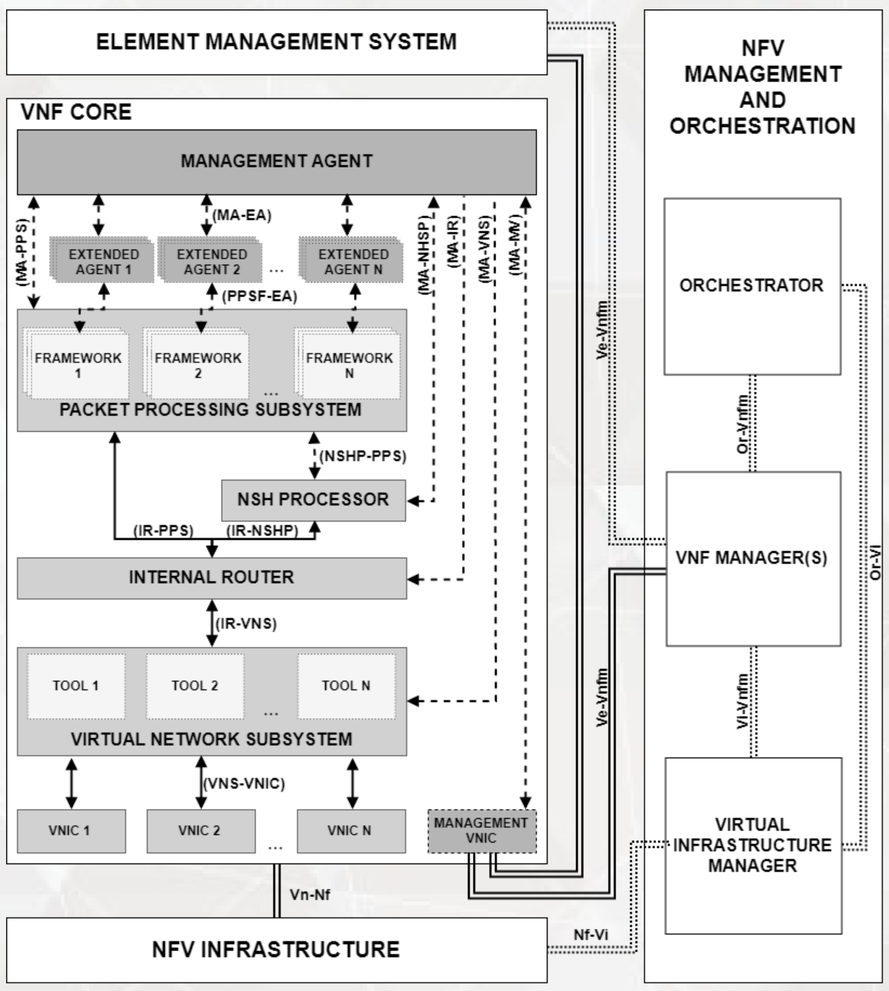

Figure 1 – VNF Platform Flexible Architecture

Each module of the architecture is designed to be loosely coupled and to present well-defined access interfaces. In this way, modules can be modified or replaced as the need appears. The modules are described below:

- **VNF Core**: it is a virtualized instance (*e.g.*, Virtual Machine, and Container) that provides the computational resources and software environment for the execution of the other modules.
- **Virtual Network Subsystem** (VNS): this module accesses the Virtual Network Interface Controllers (VNICs) for sending/receiving network packets.
- **Internal Router** (IR): the IR uses communication channels (*e.g.*, using shared memory, pipes, sockets) to forward the network packets between the VNS, the PPS, and the NSHP.
- **NSH Processor** (NSHP): this module provides an abstraction for NSH unaware NFs, acting as an NSH proxy. Specifically, when activated, the IR forwards the network traffic to be processed by the NSHP.
- **Packet Processing Subsystem** (PPS): the PPS executes the NFs on the platform. The NFs must be developed using supported frameworks.
- **Management Agent** (MA): the primary goal of an MA is to monitor and control the execution of VNFs. Furthermore, it is also responsible for coordinating the execution and configuration of all internal modules of the VNF platform.
- **Extended Agent** (EA): this module monitors a particular NF/VNFC. It is developed by the creator of the VNF/VNFC, as it acts on the individual management data of those implementations (*e.g.*, the number of discarded packets by a firewall).

CONCLUSION

This is still a work in progress, a **prototype platform***, fully compliant with the presented architecture and based on technologies such as L2 Sockets, DPDK, Click Modular Router, Python 3, and Java, is currently being developed. As for the next steps, the developed **prototype will be deployed on a testbed** reflecting a common network scenario, in order to gather performance measurements and **identify possible bottlenecks** and optimizations for the architecture.

*COVEN Platform Prototype (https://github.com/ViniGarcia/COVEN)

Latin American Student Workshop on Data Communication Networks (the authors gratefully acknowledge ACM SIGCOMM for granting this presentation)

REFERENCES

GARCIA, V. F. et al. An NSH-Enabled Architecture for Virtualized Network Function Platforms. In: **ADVANCED Information Networking and Applications**. [S.l.]: Springer, 2019. p. 376–387.

NFVISG, E. **Network Function Virtualisation (NFV): Terminology for Main concepts in NFV**. [S.l.], 2014.

_____. **Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action**. [S.l.], 2012.

QUINN, P.; ELZUR, U.; PIGNATARO, C. **Network Service Header (NSH) - RFC 8300**. [S.l.], 2018. (Request for Comments).