On the Design of a Flexible Architecture for Virtualized Network Function Platforms

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Universidade Federal de Santa Maria
SEDES SAPIENTIÆ
1960

Vinícius F. Garcia, Leonardo da C. Marcuzzo, Alexandre Huff, Lucas Bondan, Jéferson C. Nobre, Alberto Schaeffer-Filho, **Carlos R. P. dos Santos**, Lisandro Z. Granville, Elias P. Duarte

Global Communications Conference
2019

Straightforward Overview

- Introduction
- Related Works
- VNF Platform Architecture
- The COVEN Platform
- Case Study and Results
- Conclusion

Introduction

- **Network Function Virtualization (NFV)**
  - Network *softwarezation*
  - High elasticity and flexibility
  - Reduced CAPEX and OPEX
- **Virtualized Network Function (VNF)**
  - Network function virtual instances
    - Network Function + VNF Platform
  - Network Function (NF)
    - DHCP, IDS, DPI, …
  - VNF Platform
    - ClickOS, Click-on-OSv, OpenNetVM, …

How a VNF platform may be architected to perform network functions with multiple different requirements?

How a VNF platform may be architected to perform network functions with multiple different requirements?

- **Protocol**
  - Network Service Header (NSH)
  - OpenFlow

?

How a VNF platform may be architected to perform network functions with multiple different requirements?

- **Protocol**
  - Network Service Header (NSH)
  - OpenFlow
- **Network**
  - Sockets
  - Netmap
  - DPDK
  - PF_ring

?

How a VNF platform may be architected to perform network functions with multiple different requirements?

- **Protocol**
  - Network Service Header (NSH)
  - OpenFlow
- **Network**
  - Sockets
  - Netmap
  - DPDK
  - PF_ring
- **Language**
  - Click
  - VPP
  - C/C++
  - Python

?

How a VNF platform may be architected to perform network functions with multiple different requirements?

- **Protocol**
  - Network Service Header (NSH)
  - OpenFlow
- **Network**
  - Sockets
  - Netmap
  - DPDK
  - PF_ring
- **Language**
  - Click
  - VPP
  - C/C++
  - Python
- **Other**
  - Element Management System (EMS)
  - Components

Related Works

ClickOS

- Protocol: ***undefined***
- Network: ***netmap***
- Language: ***click***
- Other: ***none***

OpenNetVM

- Protocol: ***undefined***
- Network: ***DPDK***
- Language: ***NFLib***
- Other: ***none***

Click-on-OSv

- Protocol: ***undefined***
- Network: ***DPDK***
- Language: ***click***
- Other: ***native EMS***

OPNFV SampleVNF

- Protocol: ***undefined***
- Network: ***DPDK***
- Language: ***undefined***
- Other: ***none***

Related Works

ClickOS

- Protocol: *undefined*
- Network: *n*
- L
- Other: *none*

OpenNetVM

- Proto *undefined*
- Network: *DPDK*
- Langua
- Ot

n-OSv

- Protocol: *undefined*
- Ne *DK*
- Language: *click*
- Other:

##

- tocol: *undefined*
- Network: *DPDK*
- Language: *undefined*
- Other: *none*

Do not natively support important NFV features (e.g., NSH and VNFC)

Inflexible and with dependencies

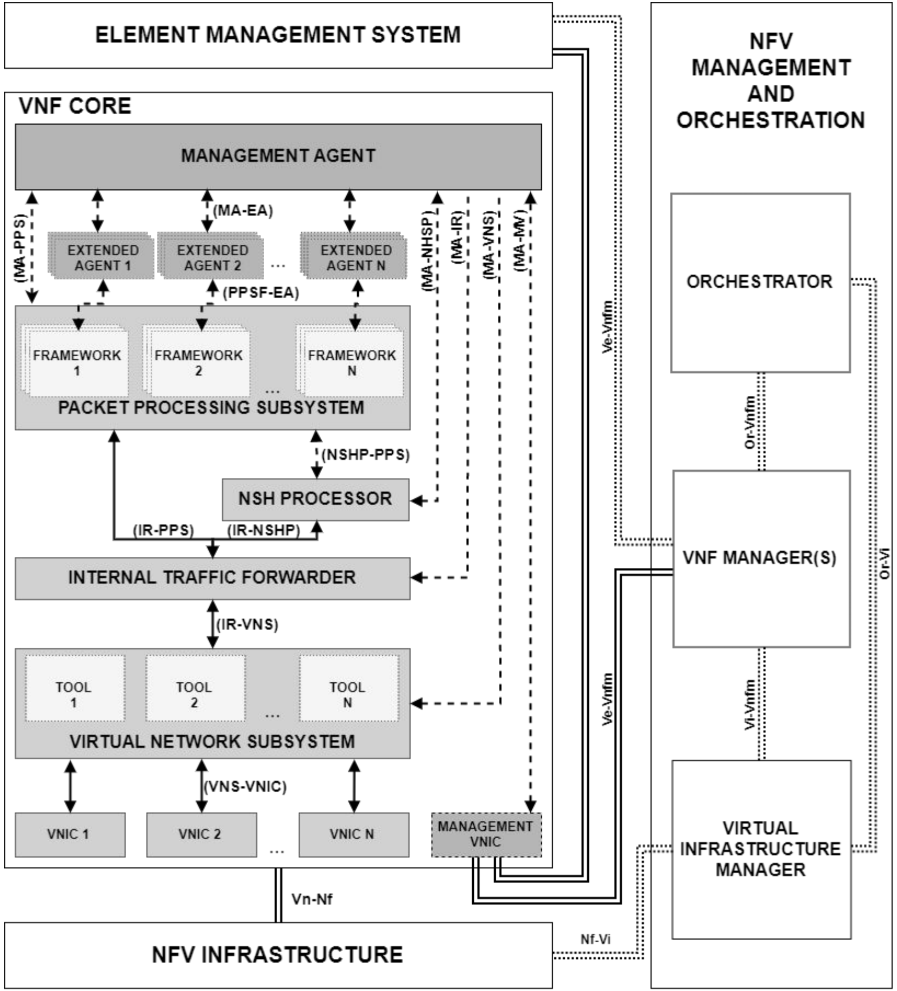
Do not follow any reference architecture (very distinct implementations)

VNF Platform Architecture

Architecture for the design and development of VNF platforms, with features such as NSH processing, VNFC deployment, internal modules dynamic traffic steering, and elastic life cycle management.

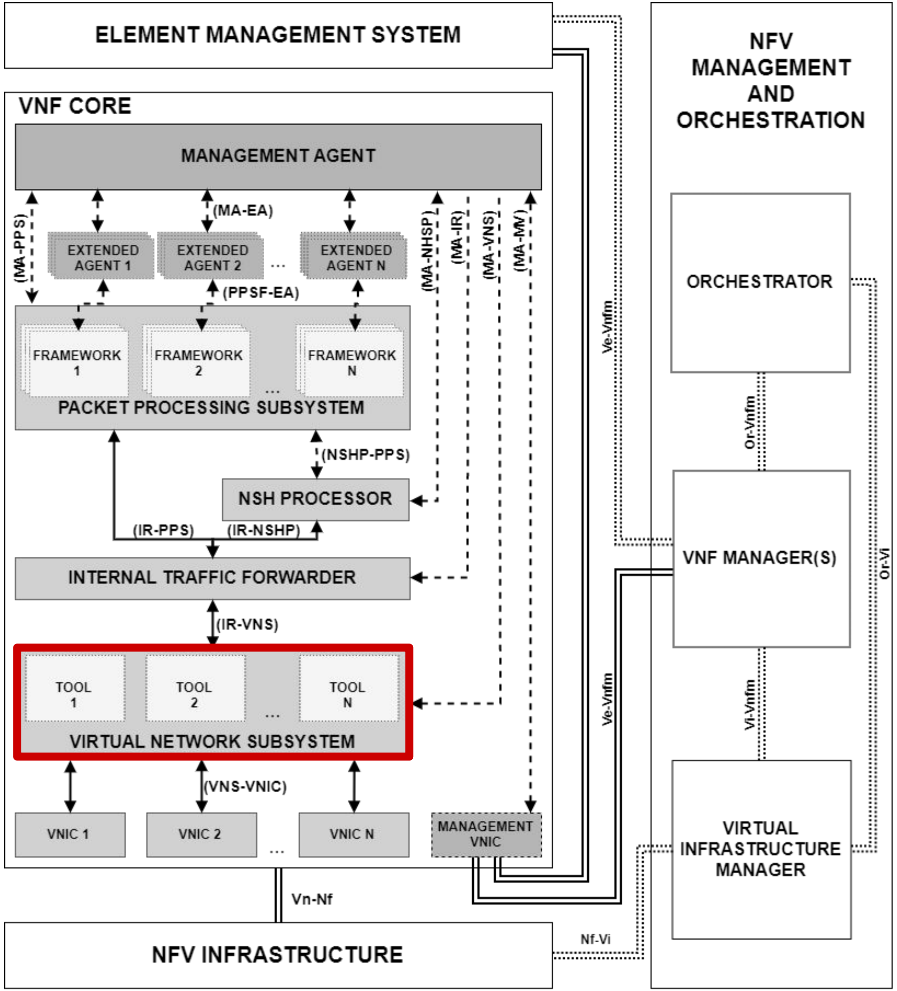
VNF Platform Architecture

- **Modularized**
  - Six modules
- **Flexible**
  - Network
  - Language
  - Protocols
  - Management
- **ETSI Compliant**
  - NFV Architecture



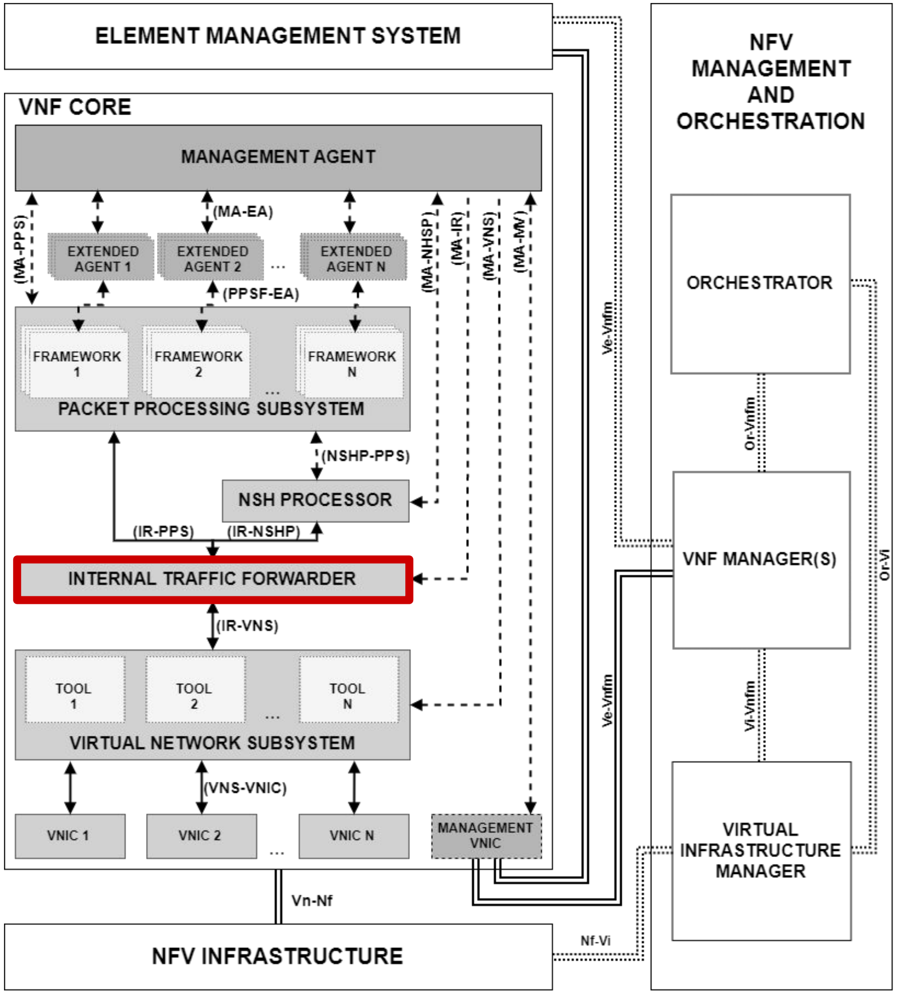
VNF Platform Architecture

- **Virtual Network Subsystem (VNS)**
  - Accesses the Virtual Network Interface Controllers (VNICs)
  - Support to multiple tools (*e.g.*, sockets, DPDK, netmap)



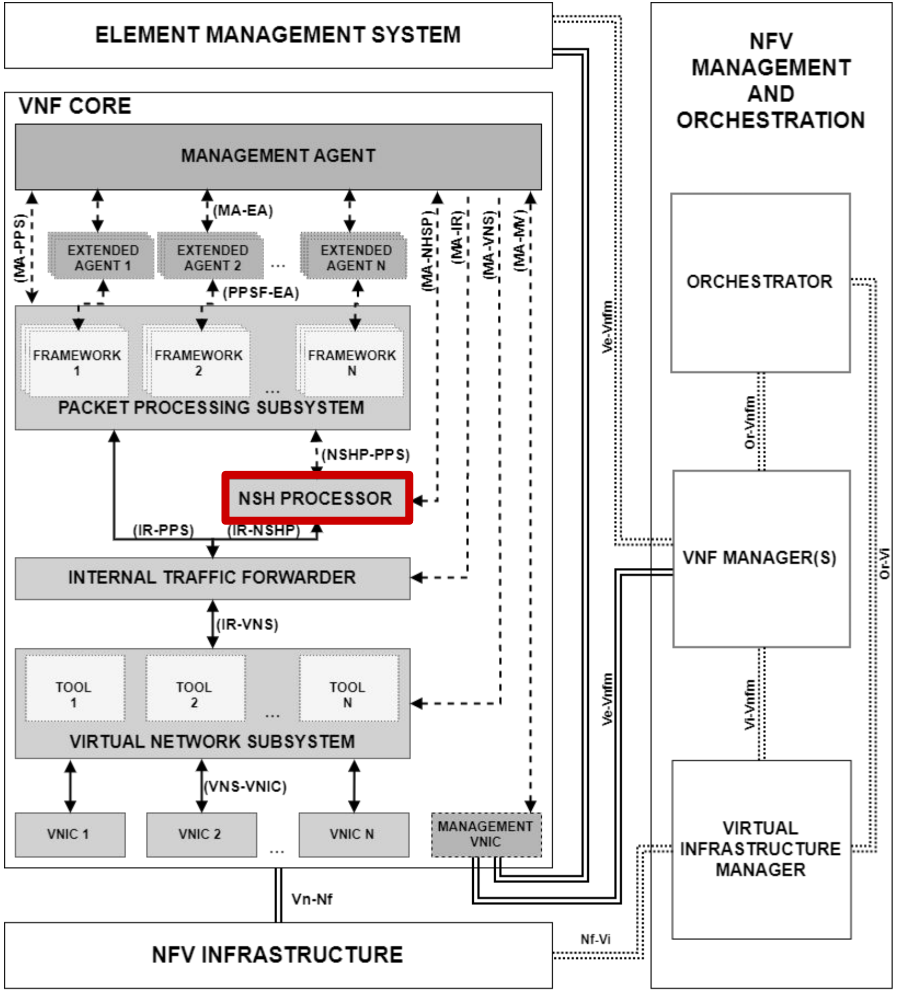
VNF Platform Architecture

- **Internal Traffic Forwarder (ITF)**
  - Receives packets from the VNS
  - Forwards packets to the components in the PPS
  - Ensures the correct packet processing order



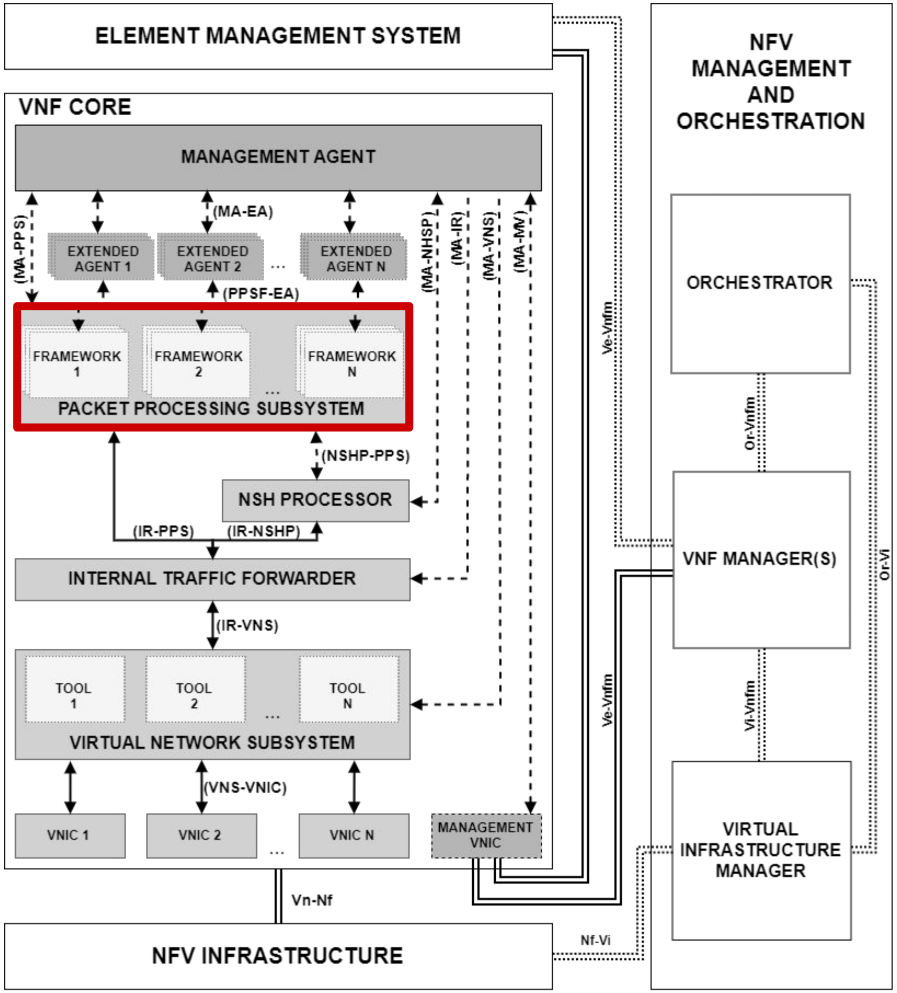
VNF Platform Architecture

- **NSH Processor**
  - NSH native internal proxy
  - Optional activation module
  - Three scenarios:
    - No NSH
    - Unaware NFs (VNFCs) + NSH
    - Aware NFs (VNFCs) + NSH



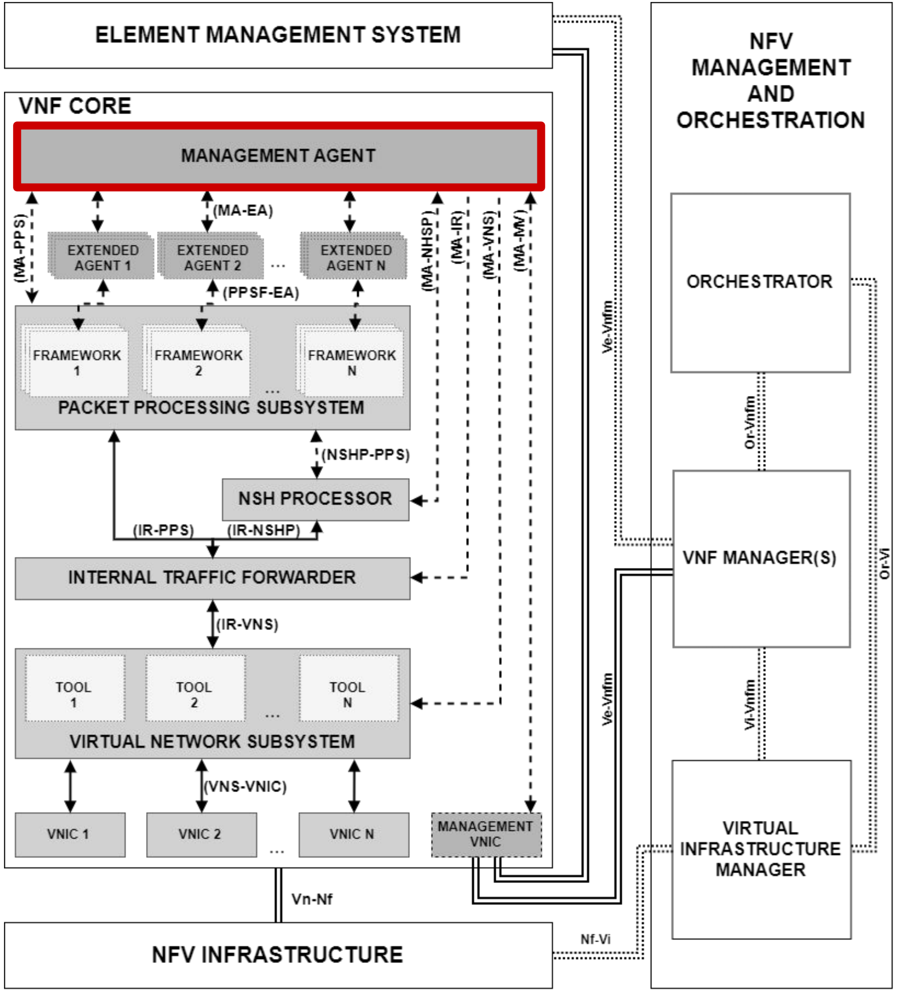
VNF Platform Architecture

- **Packet Processing Subsystem (PPS)**
  - Life cycle control of VNFCs processes
  - Processes packets received from the ITF
  - Support to multiple frameworks (*e.g.*, C, Python, Click, VPP)



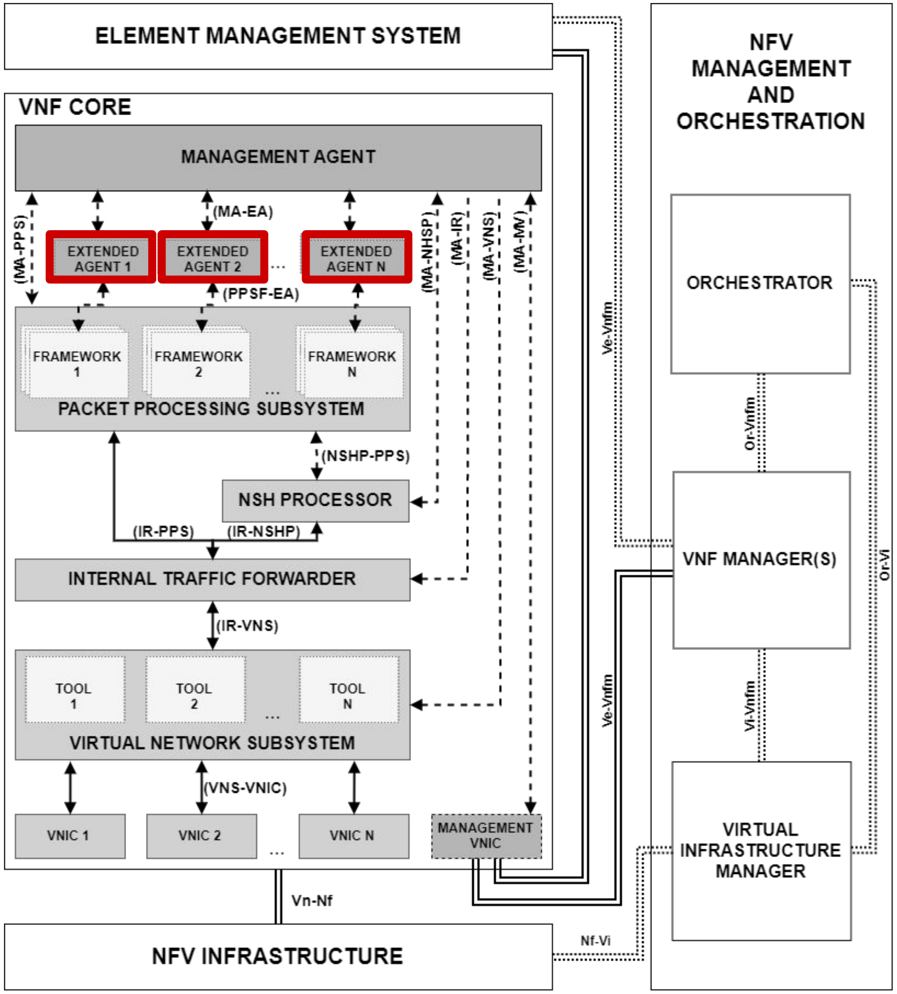
VNF Platform Architecture

- **Management Agent (MA)**
  - Parses the VNF Package (VNFP)
  - Configures and manages all the other modules
  - Interfaces the EMS, VNFM, and Network Operator



VNF Platform Architecture

- **Extended Agent (EA)**
  - Monitors/controls a particular VNFC
  - Is provided by the VNFC developer
  - Recovers specific information (*e.g.*, times that a rule of a firewall was triggered)



The COVEN Platform

- **COVEN: Proof-of-Concept Platform**
  - Compliant with the VNF platform architecture
- **Implementation Settings**
  - VNF core (base system): **Debian 8**
  - Internal modules: **Python 3**
  - Internal communication: **shared memory** and **L3 sockets**
  - Management interface: **REST**
  - Virtual Network Subsystem tools: **L2 Sockets**
  - Packet Processing Subsystem frameworks: **Click**, **C**, **Python 3**, **Java**, and **JavaScript**

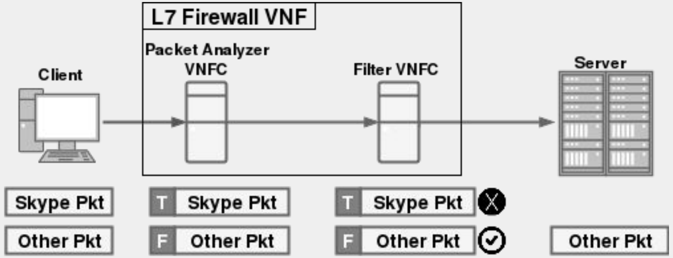
Case Study and Results

Development of a L7 firewall network function with multiple VNFC to detect and discard Skype traffic

- **Platform Validation**
  - Modules interoperation
  - Heterogeneous VNFC deployment
  - Context header (NSH) in-band control
- **Other Objectives**
  - Evaluate the RTT impacts of deployment of the same VNFC developed with different frameworks

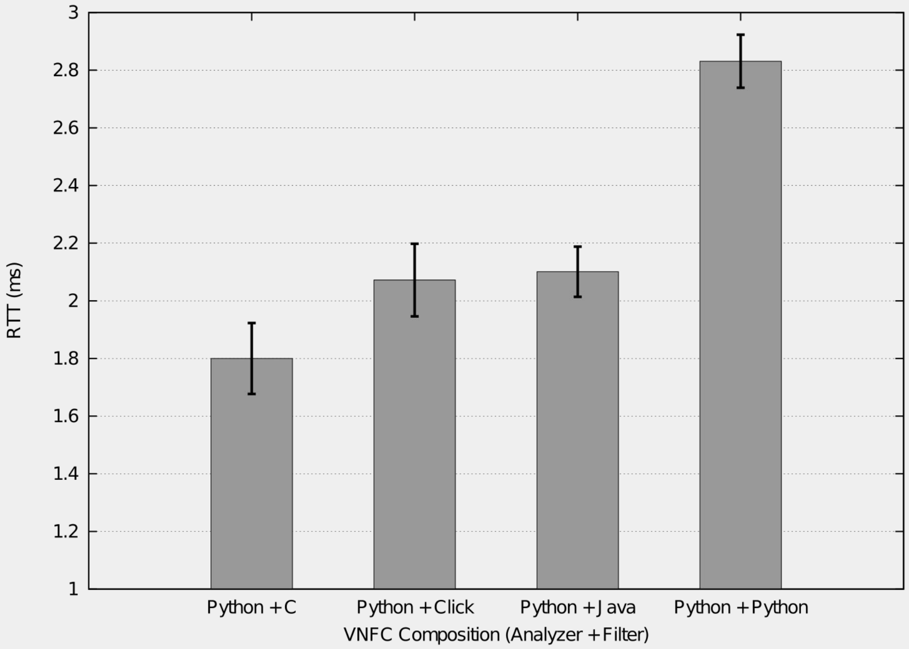
Case Study and Results

- **COVEN Setup**
  - NF: **L7 Firewall** (Skype packet blocking)
  - VNFCs: **Packet Analyzer (PA)** and **Filter (F)**
    - Packet Analyzer: **Python 3**
    - Filter: **Click**, **C**, **Java**, and **Python 3**
  - Network: **L2 Sockets**



Case Study and Results

- **RTT Tests Results**
    - The framework used to implement a VNFC impacts on the RTT
    - Processing overhead
        - Translation
        - Interpretation
        - Abstraction



Conclusion

- **VNF Platform Architecture**
  - Standard modules and communication connections
  - Support to innovative NFV features
- **COVEN Proof-of-Concept Platform**
  - Successfully executed the case study
  - Validate the architecture modules, connection, and features
- **Future Work**
  - Improvements in the COVEN platform
  - New techniques regarding VNFCs (*e.g.*, lightweight bottleneck detection, dynamic composition of NFs)

On the Design of a Flexible Architecture for Virtualized Network Function Platforms

Thanks!

Any questions?

Carlos R. P. dos Santos

csantos@inf.ufsm.br