

H2020 5GASP Project

Grant No. 101016448

## D4.3 Initial Design of NetApps in the Automotive and PPDR Verticals

### Abstract

After defining methodology for designing and developing NetApps in previously issued deliverables *D4.1 Initial Methodology for Designing and Developing NetApps* [1] and *D4.2 Final Methodology for Designing and Developing NetApps* [2], this document reports status and results of the initial phases in the process of creating automotive and PPDR-vertical NetApps achieved within the tasks *T4.2 Supporting Automotive vertical NetApps* and *T4.3 Supporting PPDR vertical NetApps*. The process of NetApp creation (i.e., design and development) involves multiple steps, as well as considerations related to the NetApp testing scenarios and requirements for the testbed facilities. The document focuses on reporting practical results of the initial NetApp creation phase, thus enhancing the discussion on the current status and achievements and presents certain sections from configuration files and screenshots. Through the results presented in the document, one may get a clearer picture from the developers' point of view on how the NetApp creation process looks like in practice,

what are potential challenges, what obstacles the developer needs to overcome, etc. The latter lead us also to the inevitable conclusion not all NetApps are at the same level of the maturity for now, however, all are progressing. As suggested by the deliverable's title, this is an initial report, while the final report on NetApp development comes in deliverable *D4.4 Development of NetApps in the Automotive and PPDR Verticals*.

## Document properties

Document number	1
Document title	D4.3 Initial Design of NetApps in the Automotive and PPDR Verticals
Document responsible	Antonio Skarmeta
Document editor	Rudolf Sušnik
Editorial team	Internet Institute
Target dissemination level	Public
Status of the document	Final version
Version	1.0

## Document history

Revision	Date	Issued by	Description
0.1	20/01/2022	ININ	Initial ToC draft
0.2	22/03/2022	ININ	Draft ready for internal review
0.3	27/03/2022	ININ	Internal review consolidated
1.0	31/03/2022	ININ	Final version

## List of Authors

Company	Name	Contribution
ININ	Luka Koršič, Rudolf Sušnik, Janez Sterle	Abstract, Introduction, NetApp #9, Conclusions
YoGoKo	Yakub Abualhoul	NetApp Design methodology, NetApp #2, NetApp #3
OdinS	Jorge Gallego Madrid	NetApp #1, NetApp #4, NetApp Design methodology
Lamda Networks	Leonidas Lymberopoulos, Angeliki Kavvalou	NetApp #8
UnivBris	Navdeep Uniyal, Xenofon Vasilakos	NetApp #7
VMware	Vesselin Arnaudov	NetApp Design methodology

<b>BLB</b>	Roman Odarchenko	NetApp #5, NetApp #6
<b>DriveU</b>	Eli Shapira	NetApp #5, NetApp #6
<b>Neobility</b>	Andrei Radulescu	NetApp #10
<b>UoP</b>	Christos Tranoris, Kostis Trantzas, Ioannis Chatzis	NetApp Design methodology, NetApp #11

## Disclaimer

This document has been produced in the context of the 5GASP Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement number 101016448.

All information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The reader thereof uses the information at its sole risk and liability.

For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the author's view.

# Contents

<b>ABSTRACT .....</b>	<b>1</b>
<b>DOCUMENT PROPERTIES .....</b>	<b>2</b>
<b>DOCUMENT HISTORY.....</b>	<b>2</b>
<b>LIST OF AUTHORS .....</b>	<b>2</b>
<b>DISCLAIMER.....</b>	<b>3</b>
<b>CONTENTS .....</b>	<b>4</b>
<b>LIST OF FIGURES.....</b>	<b>5</b>
<b>LIST OF TABLES .....</b>	<b>7</b>
<b>LIST OF ACRONYMS .....</b>	<b>8</b>
<b>DEFINITIONS .....</b>	<b>10</b>
<b>1. INTRODUCTION .....</b>	<b>11</b>
1.1. OBJECTIVES OF THIS DOCUMENT .....	11
1.2. DOCUMENT STRUCTURE.....	12
<b>2. NETAPP DESIGN METHODOLOGY .....</b>	<b>13</b>
2.1. NETAPP LIFECYCLE.....	13
2.2. 5GASP ARCHITECTURE OVERVIEW .....	14
2.3. DESIGN PHASE OVERVIEW.....	14
2.4. VM-BASED AND CONTAINER-BASED NETAPPS .....	15
<b>3. NETAPPS INITIAL DESIGN REPORT .....</b>	<b>16</b>
3.1. NETAPP #1 .....	16
3.1.1 <i>NetApp functional overview, current status and next steps .....</i>	16
3.1.2 <i>NetApp analysis .....</i>	18
3.1.3 <i>NetApp definition.....</i>	20
3.1.4 <i>NetApp's tests plan and definition.....</i>	24
3.1.5 <i>NetApp requirements for testbeds.....</i>	25
3.2. NETAPP #2 .....	25
3.2.1 <i>NetApps functional overview, current status and next steps.....</i>	25
3.2.2 <i>NetApp analysis .....</i>	26
3.2.3 <i>NetApp definition.....</i>	35
3.2.4 <i>NetApp's tests plan and definition.....</i>	39
3.2.5 <i>NetApp requirements for testbeds.....</i>	40
3.3. NETAPP #3 .....	40
3.3.1 <i>NetApp requirements for testbeds.....</i>	40
3.3.2 <i>NetApp analysis .....</i>	41
3.3.3 <i>NetApp definition.....</i>	42
3.3.4 <i>NetApp's tests plan and definition.....</i>	44
3.3.5 <i>NetApp requirements for testbeds.....</i>	45
3.4. NETAPP #4 .....	45
3.4.1 <i>NetApp functional overview, current status and next steps .....</i>	45
3.4.2 <i>NetApp analysis .....</i>	46
3.4.3 <i>NetApp definition.....</i>	47
3.4.4 <i>NetApp's tests plan and definition.....</i>	48
3.4.5 <i>NetApp requirements for testbeds.....</i>	49
3.5. NETAPP #5 .....	49

3.5.1	<i>NetApp functional overview, current status and next steps</i> .....	49
3.5.2	<i>NetApp analysis</i> .....	51
3.5.3	<i>NetApp definition</i> .....	52
3.5.4	<i>NetApp's tests plan and definition</i> .....	54
3.5.5	<i>NetApp requirements for testbeds</i> .....	56
3.6.	NETAPP #6 .....	56
3.6.1	<i>NetApp functional overview, current status and next steps</i> .....	56
3.6.2	<i>NetApp analysis</i> .....	58
3.6.3	<i>NetApp definition</i> .....	60
3.6.4	<i>NetApp's tests plan and definition</i> .....	63
3.6.5	<i>NetApp requirements for testbeds</i> .....	64
3.7.	NETAPP #7 .....	65
3.7.1	<i>NetApp functional overview, current status and next steps</i> .....	65
3.7.2	<i>NetApp analysis</i> .....	67
3.7.3	<i>NetApp definition</i> .....	68
3.7.4	<i>NetApp's tests plan and definition</i> .....	70
3.7.5	<i>NetApp requirements for testbeds</i> .....	71
3.8.	NETAPP #8 .....	72
3.8.1	<i>NetApp functional overview, current status and next steps</i> .....	72
3.8.2	<i>NetApp analysis</i> .....	76
3.8.3	<i>NetApp definition</i> .....	78
3.8.4	<i>NetApp's tests plan and definition</i> .....	80
3.8.5	<i>NetApp requirements for testbeds</i> .....	81
3.9.	NETAPP #9 .....	81
3.9.1	<i>NetApp functional overview, current status and next steps</i> .....	81
3.9.2	<i>NetApp analysis</i> .....	87
3.9.3	<i>NetApp definition</i> .....	90
3.9.4	<i>NetApp's tests plan and definition</i> .....	94
3.9.5	<i>NetApp requirements for testbeds</i> .....	96
3.10.	NETAPP #10 .....	97
3.10.1	<i>NetApp functional overview, current status and next steps</i> .....	97
3.10.2	<i>NetApp analysis</i> .....	98
3.10.3	<i>NetApp definition</i> .....	99
3.10.4	<i>NetApp's tests plan and definition</i> .....	101
3.10.5	<i>NetApp requirements for testbeds</i> .....	102
3.11.	NETAPP #11 .....	102
3.11.1	<i>NetApp functional overview, current status and next steps</i> .....	102
3.11.2	<i>NetApp analysis</i> .....	104
3.11.3	<i>NetApp definition</i> .....	106
3.11.4	<i>NetApp's tests plan and definition</i> .....	108
3.11.5	<i>NetApp requirements for testbeds</i> .....	108
4.	CONCLUSIONS .....	109
	BIBLIOGRAPHY .....	110

## List of Figures

FIGURE 2.1: 5GASP NETAPP LIFECYCLE .....	13
FIGURE 2.2: PHASES OF THE DESIGN OF A NETAPP IN THE CONTEXT OF 5GASP .....	15
FIGURE 3.1.1: VOBU NETAPP ARCHITECTURE .....	16
FIGURE 3.1.2: VOBU NSD INSTANTIATED FROM OSM10 .....	17
FIGURE 3.1.3: VOBU VNFS INSTANCES IN OSM10 DASHBOARD .....	17
FIGURE 3.1.4: VOBU INSTANCES AS SEEN IN THE LOCAL OPENSTACK .....	17
FIGURE 3.1.5: SIMULATED PHYSICAL OBU REQUESTING FOR A VOBU .....	18

FIGURE 3.1.6: VOBU MANAGER RESPONDING TO A VOBU REQUEST AND ASSIGNING IT TO THE SIMULATED OBU .....	18
FIGURE 3.2.1: VRSU NETAPP (NETAPP #2) AND C-ITS-S NETAPP (NETAPP #3) GENERAL ARCHITECTURE.....	26
FIGURE 3.2.2: DATA FLOWS WITH EXTERNAL NETAPP AS THE DATA SOURCE .....	27
FIGURE 3.2.3: DATA FLOWS BETWEEN NETAPP #3 LOCAL PROCESSING MODULE AS THE DATA SOURCE.....	29
FIGURE 3.2.4: DATA FLOWS NETAPP #2 LOCAL PROCESSING MODULE AS THE DATA SOURCE .....	30
FIGURE 3.2.5: DATA FLOWS V-ITS-S AS THE DATA SOURCE.....	31
FIGURE 3.2.6: DATA FLOWS V-ITS-S TO NETAPP #2 THROUGH SHORT-RANGE INTERFACES.....	32
FIGURE 3.2.7: DATA FLOWS V-ITS-S TO NETAPP #2 THROUGH 4G/5G INTERFACES .....	33
FIGURE 3.4.1: MULTI-DOMAIN MIGRATION NETAPP ARCHITECTURE .....	45
FIGURE 3.5.1: HIGH-LEVEL ARCHITECTURE OF VEHICLE-TO-CLOUD REAL-TIME COMMUNICATION (V2C R2C).....	50
FIGURE 3.5.2: JENKINS BUILDER SNAPSHOT .....	50
FIGURE 3.6.1: REMOTE HUMAN DRIVING NETAPP - TELEOPERATION FOR ASSISTING VEHICLES IN COMPLEX SITUATIONS .....	57
FIGURE 3.6.2: VEHICLE-TO-CLOUD REAL-TIME COMMUNICATION (V2C R2C) NETAPP .....	57
FIGURE 3.7.1: EFFICIENT MEC HANDOVER NETAPP .....	65
FIGURE 3.7.2: EMHO ON-BOARDED IN OSM .....	66
FIGURE 3.7.3: INSTANCE OF VNF IN OPENSTACK .....	66
FIGURE 3.7.4: RELEVANT NETWORKS IN OPENSTACK .....	66
FIGURE 3.7.5: EMHO - NSD AND VNFD RELATIONS .....	68
FIGURE 3.7.6: EMHO - TESTING PROCEDURE.....	71
FIGURE 3.8.1: PRIVACYANALYZER'S DEPLOYMENT IN THE ININ AND UOP TESTBEDS WITH THE NETAPP ANALYZING THE PRIVACY STRENGTH OF NETWORKD MESSAGES FROM 5G IOPS UEs IN THE PPDR USE .....	72
FIGURE 3.8.2: LOCAL-OSM ONBOARDING OF THE PRIVACYANALYZER DEMO.....	73
FIGURE 3.8.3: PRIVACYANALYZER USER INTERFACE SHOWING THE OUTPUT OF OUR DEMO SESSION .....	74
FIGURE 3.8.4: PRIVACYANALYZER MESSAGE 1 .....	74
FIGURE 3.8.5: PRIVACYANALYZER RUNTIME LOGS FROM BACKEND COMPONENTS (LEFT) AND WEB SERVICES TO UI (RIGHT) .....	75
FIGURE 3.8.6: PRIVACYANALYZER COMPONENTS AND DATA FLOW.....	77
FIGURE 3.9.1: REGULAR DAY-TO-DAY OPERATION PHASE, WHERE THE NETAPP IS DEPLOYED IN THE DISTRIBUTED MANNER, INTERNET ACCESS AND 5G SERVICES ARE PROVIDED TO PDDR USERS FROM THE PUBLIC 5G CORE NETWORK (ALTHOUGH MOBILE CORE VNF IS DEPLOYED AT THE MEC, IT IS NOT ACTIVE IN THIS CASE).....	82
FIGURE 3.9.2: ISOLATED OPERATIONS MODE FOLLOWING UPLINK FAILURE DETECTION (BASED ON THE HEALTH-CHECK MECHANISM IMPLEMENTED) AND THE RECONFIGURATION OF THE CLOUD BBU VNF TO USE THE MOBILE CORE VNF PROVISIONED ON THE EDGE/MEC, ENABLING BASIC/PRIVATE 5G SERVICES TO THE USERS.....	82
FIGURE 3.9.3: USING "DOCKER RUN" COMMAND TO CREATE A WRITEABLE CONTAINER LAYER OVER THE SPECIFIED IMAGES OF CN AND GNB, AND STARTING THEM .....	83
FIGURE 3.9.4: REQUIRED NETAPP PROCESSES ARE UP AND RUNNING .....	83
FIGURE 3.9.5: ENV SAMPLE FILE.....	84
FIGURE 3.9.6: DETAILS ON SWITCHING BETWEEN THE TWO CORES, I.E., SWITCHING FROM DISTRIBUTED TO STANDALONE MODE....	85
FIGURE 3.9.7: GRAFANA DASHBOARD SHOWING RADIO SETTINGS FOR THE 5G UE CONNECTED TO 5G IOPS SLICE .....	85
FIGURE 3.9.8: GRAFANA DASHBOARD SHOWING AVERAGE DOWNLOAD SPEED FOR THE 5G UE CONNECTED TO 5G IOPS SLICE.....	85
FIGURE 3.9.9: DEPLOYMENT OF CN VNF/VM AND IOPS VNF/VN IN OSM10. ....	86
FIGURE 3.9.10: DEPLOYMENT OF CN NS AND IOPS NS IN OSM10 .....	86
FIGURE 3.9.11: DAY-1 CONFIGURATION FOR THE 5G IOPS SERVICE IN OSM10 .....	86
FIGURE 3.9.12: ACTIONS/COMMANDS AVAILABLE IN DAY-1/DAY-2 CONFIGURATION OF THE 5G IOPS NETAPP.....	87
FIGURE 3.9.13: RELATIONS BETWEEN NSDS AND VNFS.....	89
FIGURE 3.9.14: QMON MONITORING TOOL SYSTEM MODULES .....	94
FIGURE 3.9.15: INTEGRATION OF END-TO-END NETAPP PERFORMANCE TESTING IN ININ'S TESTBED USING QMON MONITORING TOOL.....	95
FIGURE 3.9.16: LOG FILE CONTAINS RESULTS OF IPERF AND RADIO TESTS .....	96
FIGURE 3.10.1: HIGH LEVEL ARCHITECTURE DIAGRAM OF THE NEOBUS NETAPP .....	97
FIGURE 3.10.2: "DOCKER RUN" COMMANDS TO CREATE THE DOCKER CONTAINERS .....	98
FIGURE 3.11.1: HIGH LEVEL ARCHITECTURE DIAGRAM OF THE FIDEGAD NETAPP.....	102
FIGURE 3.11.2: ARCHITECTURAL DIAGRAM OF THE FIDEGAD NETAPP .....	103
FIGURE 3.11.3: NS INSTANTIATION COMMAND WITH DAY-1 PARAMETERS .....	104
FIGURE 3.11.4: RUNNING NS INSTANTIATED WITH DAY-1 PARAMETERS.....	104
FIGURE 3.11.5: RUNNING KUBERNETES SERVICES.....	104
FIGURE 3.11.6: RUNNING KUBERNETES DEPLOYMENTS .....	104

## List of Tables

TABLE 3.1.1: VOBU NSD .....	19
TABLE 3.1.2: VNF OBU MANAGER .....	19
TABLE 3.1.3: VNF AGGREGATOR .....	19
TABLE 3.1.4: VNF vOBUs .....	19
TABLE 3.1.5: VOBU NETAPP's NEST .....	20
TABLE 3.2.1: VRSU NSD .....	34
TABLE 3.2.2: VRSU MANAGER VNF .....	34
TABLE 3.2.3: VRSU VNF .....	34
TABLE 3.2.4: VRSU NETAPP (NETAPP #2) NEST .....	36
TABLE 3.2.5: SERVICE TEST EXAMPLES FOR THE C-ITS NETAPPS (NETAPP #2) .....	40
TABLE 3.3.1: VIRTUAL COOPERATIVE ITS-S NETAPP NSD .....	41
TABLE 3.3.2: VIRTUAL COOPERATIVE ITS-S NETAPP VNF .....	42
TABLE 3.3.3: C-ITS NETAPP (NETAPP #3) NEST .....	43
TABLE 3.3.4: SERVICE TEST EXAMPLES FOR THE C-ITS NETAPPS (NETAPP #3) .....	44
TABLE 3.4.1: MULTI DOMAIN MIGRATION NSD .....	46
TABLE 3.4.2: VOBU VNF .....	46
TABLE 3.4.3: DISTRIBUTED DB VNF .....	47
TABLE 3.4.4: MIGRATE CONTROLLER VNF .....	47
TABLE 3.4.5: MULTI-DOMAIN NETAPP's NEST .....	48
TABLE 3.5.1: CLOUD NSD .....	51
TABLE 3.5.2: VEHICLE NSD .....	51
TABLE 3.5.3: VEHICLE VNF .....	52
TABLE 3.5.4: SERVER VNF .....	52
TABLE 3.5.5: V2C R2C NETAPP's NEST .....	53
TABLE 3.6.1: CLOUD NSD .....	58
TABLE 3.6.2: VEHICLE NSD .....	58
TABLE 3.6.3: CLOUD NSD .....	59
TABLE 3.6.4: VNF VEHICLE SIDE .....	59
TABLE 3.6.5: VNF SERVER .....	59
TABLE 3.6.6: VNF NODE .....	59
TABLE 3.6.7: V2C R2C NETAPP's NEST .....	60
TABLE 3.7.1: EMHO NSD .....	67
TABLE 3.7.2: VNF EMHO ML .....	67
TABLE 3.7.3: EMHO NETAPP's NEST .....	68
TABLE 3.8.1: PRIVACY ANALYZER MEC NSD .....	76
TABLE 3.8.2: PRIVACY ANALYZER CORE NSD .....	76
TABLE 3.8.3: PRIVACY ANALYZER MEC VNF .....	76
TABLE 3.8.4: PRIVACY ANALYZER CORE VNF .....	77
TABLE 3.9.1: BBU NSD .....	87
TABLE 3.9.2: CN NSD .....	88
TABLE 3.9.3: IOPS (CN+BBU) NSD .....	88
TABLE 3.9.4: 5G IOPS BBU NS .....	88
TABLE 3.9.5: 5G IOPS MOBILE CORE NS .....	88
TABLE 3.9.6: 5G IOPS CN+BBU NS .....	89
TABLE 3.9.7: 5G IOPS NETAPP's NEST .....	90
TABLE 3.10.1: VRO NSD .....	98
TABLE 3.10.2: VNF USER API .....	98
TABLE 3.10.3: VNF DRIVER API .....	99
TABLE 3.10.4: VNF ROUTING ENGINE .....	99
TABLE 3.10.5: VRO NETAPP's NEST .....	100
TABLE 3.11.1: CAMERAALISTR NSD .....	105

TABLE 3.11.2: CAMERAAISTR CLIENT NSD.....	105
TABLE 3.11.3: CAMERAAISTR VNF.....	105
TABLE 3.11.4: CAMERAAISTR CLIENT VNF.....	105
TABLE 3.11.5: FIDEGAD NETAPP'S NEST .....	106

## List of Acronyms

5GASP	5G Application & Services experimentation and certification Platform
AMF	Access and Mobility Management Function
AI	Artificial Intelligence
API	Application Programming Interface
ARFCN	Absolute Radio-Frequency Channel Number
AWS	Amazon Web Services
BBU	Base-band Unit
BW	Bandwidth
CI/CD	Continuous Integration and Continuous Deployment
C-ITS-S	Cooperative Intelligent Transport Systems Station
CN	Core Network
CNF	Cloud-native Network Function
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
C-V2X	Cellular Vehicle to Everything
D2D	Device-to-Device
DevOps	Development and Operations
DL	Download
DNS	Domain Name Service
DQT	Data Quality Testing
DRT	Demand Responsive Transport
DSP	Digital Services Provider
EMHO	Efficient MEC Handover
eNB	Evolved NodeB (=LTE/4G Base Station)
FTP	File Transfer Protocol
gNB	gNodeB (= 5G Base Station)
GPS	Global Positioning System
GST	Generic Network Slice Template
GTP	General Tunneling Protocol
HDD	Hard Disk Drive
HW	Hardware
HTTP(S)	HyperText Transfer Protocol (Secure)
ICMP	Internet Control Message Protocol
IOPS	Isolated Operations for Public Security
IoT	Internet of Things
IPv6	Internet Protocol version 6
ITS	Intelligent Transport Systems
ITS-S	Intelligent Transport Systems Station

JSON	JavaScript Object Notation
K8s	Kubernetes
KNF	Kubernetes Native Function
KPI	Key Performance Indicator
MANO	MANagement and Orchestration
MC	Mission-Critical
MCC	Mobile Country Code
MEC	Multi-access Edge Computing
MNC	Mobile Network Code
MNO	Mobile network operator
OBU	On-Board Unit
OSM	Open-Source MANO
NEST	Network Slice Type
NetApps	Network Applications
NFV	Network Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	NFV Orchestrator
NS	Network Services
NSA	Non-Standalone
NSD	Network Service Descriptor
NSP	Network Service Provider
NR	New Radio
OBU	On-Board Unit
OSM	Open-source MANO
PII	Personal Identifiable Information
PLMN	Public Land Mobile Network
PLR	Packet Loss Ratio
PNFD	Physical Network Function Descriptor
PoC	Proof of Concept
PoP	Point of Presence
PPDR	Public Protection and Disaster Relief
R2C	Roadside-to-Cloud
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
REST	REpresentational State Transfer
R-ITS-S	Road-side Intelligent Transport System Station
RO	Resource Orchestrator
RQT	Radio Quality Testing
RSU	Road-Side Unit
RT	Real-Time
RTSP	Real-Time Streaming Protocol
SA	Standalone
SDR	Software Defined Radio
SNMP	Simple Network Management Protocol
SMEs	Small and Medium-sized Enterprises

SDN	Software Defined Network
UE	User Equipment
UL	Upload
URL	Uniform Resource Locator
V&V	Validation and Verification
V2C/C2V	Vehicle-to-Cloud / Cloud-to-Vehicle
V2X	Vehicle-to-everything
V-ITS-S	Vehicle Intelligent Transport Systems Station
VIM	Virtual Infrastructure Management
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VOBU	Virtual On-Board Unit
vRSU	Virtual Road-Side Unit
WP	Work Package

## Definitions

This document contains specific terms to identify elements and functions that are considered to be mandatory, strongly recommended or optional. These terms have been adopted for use similar to that in IETF RFC2119 and have the following definitions:

- **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

# 1. Introduction

## 1.1. Objectives of this document

Previous deliverables *D4.1 Initial Methodology for Designing and Developing NetApps* [1] and *D4.2 Final Methodology for Designing and Developing NetApps* [2] have described in detail the methodology NetApp developers are following while designing and developing their NetApps. First practical outcomes of the design process are, for each single NetApp, reported and discussed in this deliverable. In an attempt to make the deliverable genuine in reporting the progress, it also includes certain sections of configuration files.

The deliverable is primarily focused on reporting the status of creating NetApps, therefore its content follows key methodology topics introduced in D4.1 and D4.2. To better understand the goal of each NetApp, a functional overview is presented first, followed by a short description on the status and next steps regarding the NetApp creation process toward its final stage and demonstration within the 5GASP infrastructure. Since all NetApps are not exactly at the same level of the development, certain differences between them may be observed.

Within the NetApp analysis section, a list of Network Service Descriptors (NSD) is presented for each Network Service used in a particular NetApp. Similarly, a list of Virtual Network Functions (VNFs), which implements already defined Network Services, is presented as well. The same section also depicts relations and dependencies among NSDs and VNFs.

The objective of the NetApp definition section is to present NetApps' specifics related to the packaging, health-check, or lifecycle hooks, 5G system dependencies, operation aspects and possible dependencies to other NetApps. Then, based on NetApp's Network slice type (NEST), definitions (configurations) of NSDs and VNFs are given.

Regarding the testing, validation, and certification phases defined in detail within WP5, the deliverable includes a section on NetApp test plan and definition, where both selected WP5 [3] defined test cases and individual test cases that will be performed for a NetApp are listed. In case individual test cases are envisioned for the particular NetApp, the test procedure is discussed as well.

Finally, NetApp requirements for hosting testbeds are presented for the testbed facility owners to have a better insight what capabilities are required to support a certain NetApp. Although it is expected all testbeds meet most of the requirements, there still might be some specifics required by certain NetApps.

As mentioned in the beginning, NetApp design and development outcomes reported are, where appropriate, further outlined by showing sections from the configuration files or screenshots showing NetApps' certain functionalities execution. Please note that it is necessary to zoom-in the document for some screenshots to be clearly readable. Unfortunately, this works for electronic version of the document only, but not for the printed one.

## 1.2. Document structure

The deliverable is organized into four sections. First section, i.e., Introduction, provides a brief insight into the objectives and the content of the document. Then, in section 2, a short overview of the NetApp design methodology is presented – it merely summarizes main topics from the previously issued D4.1 [1] and D4.2 [2] where the methodology is defined in the detail. The aim of section 2 is to get the reader acquainted with the 5GASP NetApp design and development methodology or to refresh the reader's knowledge in case she/he is familiar with other 5GASP deliverables. Section 3 is the main part of the deliverable, reporting each single NetApp's design and development status in details. Finally, section 4 concludes the deliverable.

## 2. NetApp design methodology

This section provides a brief overview of the design methodology, as previously specified in D4.1 [1] and D4.2 [2], to highlight key factors of the design process. Therefore, the subsection for NetApp Lifecycle is presented to better understand the position of the phase 5GASP NetApps are currently at, 5GASP architecture overview highlights the systemic view to the NetApp and its role within the 5GASP ecosystem, while the Design phase overview summarizes design phases and key considerations NetApp developers need to take into account. The section also addresses deployment methods as being an important information for the testbeds. All key points mentioned in this section are addressed in section 3 where each single NetApp design progress is reported.

The reader, already acquainted with the 5GASP's NetApp design methodology, may skip this section.

### 2.1. NetApp Lifecycle

NetApps in the lifecycle ecosystem are envisioned to have a lifecycle similar to other software services, consisting of an iterative process of design, development, testing, and validation.

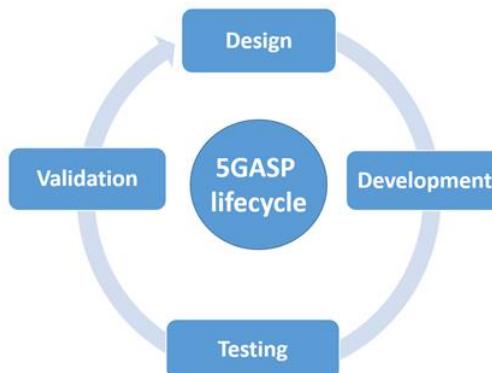


Figure 2.1: 5GASP NetApp lifecycle

As discussed further in this document, design of a NetApp consists of three main steps. The programmer must complete these steps so that their existing code can be onboarded as a NetApp through the 5GASP platform and tested through 5GASP's testing framework. Following the design phase, the development consists of materializing the design considerations studied in the previous step, resulting in, in the end, a NetApp ready to be onboarded to the 5GASP portal. Consequently, the development of the NetApp is also divided into three steps, one for each component of the needed triplet: the VNFDs/NSDs, the network slice and the test descriptors. The current version of the testing stage aims to check whether the developed NetApp matches expected requirements and to ensure that NetApp is defect-free. This process assures quality and optimized efficiency. After the testing phase Validation process takes place, the validation can be defined as the process of evaluating developed NetApp during or at the end of the development process to determine whether it satisfies specified requirements (defined KPIs).

## 2.2. 5GASP Architecture Overview

The main objective of 5GASP project, in relation to NetApps ecosystem, is the provision of a single-entry point to the overall experimentation platform. This adoption envisages to introduce an abstraction layer between the developer, or the NetApp community stakeholder, in general, and the underlying implementing infrastructure. To achieve this goal, 5GASP overall facility shall be viewed as a single administrative domain from the NetApp developer's perspective, despite being composed of several state-of-the-art interworking sites, spanning across different geographic locations. Apart from the infrastructure hosting the NetApp, 5GASP aims to establish a seamless CI/CD methodology so as specific benchmark or even custom test use cases are executed against the onboarded NetApps in an environment as closer-to-production as possible.

To achieve the above, 5GASP project proposes a unified model, based on major standards definition bodies in the telecommunications world, that has a model of a “triplet”. Each entity of the said triplet, i.e., NetApp artefacts, hosting network slice, selected test suite aggregates the overall input that is expected from the developer to a meaningful orchestration-wise bundle. Specifically, the model elected to express each entity is TMF's Service Specification [4], which provides a service-level description on topology and expected behavior. Exhaustive analysis on the designated triplet is omitted, as it is out of this document's scope, and can be found in D3.1 [5]. In addition, the adoption of a unified, widely accepted resource model ensures that the same behavior can be expected on any NFV/3rd Generation Partnership Project (3GPP) compliant 5G system, besides the 5GASP facility.

Also, 5GASP offers two additional portals, i.e., NetAppCommunity and NetAppStore, in the pursuit of the goal to lay the foundations of a 5G NetApps ecosystem. Our vision on providing a community ecosystem is of outmost importance, where its members can exchange insight through opinion sharing tools on development and testing mechanisms, thus promoting a tight support and information sharing body. On the other hand, a NetAppStore showcasing a number of innovative NetApps, from the project's 7 SMEs, that have been deployed, tested and validated onto the 5GASP open platform, should provide the guidelines and best practices on NetApp development proposing a template to build upon.

## 2.3. Design phase overview

The design phases of a NetApp in the context of 5GASP were initially proposed in D4.2 [2]. The project design methodology identified three main design phases for NetApp developers to properly onboard their NetApps and perform the tests through 5GASP's testing framework.

**Phase-I Analysis Phase:** Perform the NetApp analysis regarding NSDs, VNFs, Networks requirements as well as any vertical and use-case dependencies. This also covers the identification of the main components of the NetApp in terms of services (described with Network Service Descriptors-NSDs) and networking service components, that is, Virtual Network Functions (VNFs) in the 5G terminology. Moreover, the first phase in designing NetApp also requires early identification of the dependencies from any other NetApps.

**Phase-II Requirements Phase:** Provide a NetApp complete definition of NSDs for all associated services and define the 5G network requirements of the NetApps' slices associated with the NetApp to be instantiated on the 5GASP platform.

**Phase-III Testing Phase:** Define the corresponding set of tests and the related descriptors to avoid any instantiation and/or operation error, while onboarded on the relevant 5GASP testbed(s).

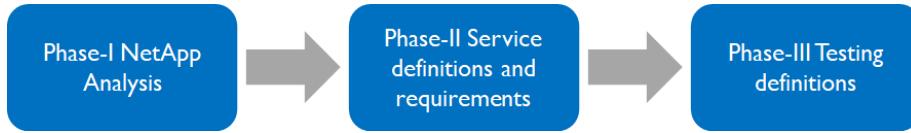


Figure 2.2: Phases of the design of a NetApp in the context of 5GASP

## 2.4. VM-based and Container-based NetApps

The NFV concept is conceived to be completely agnostic on how VNFs are instantiated, but it has to be considered when planning a deployment. The most common methods to deploy them is as Virtual Machines (VMs) or as containers. Although the VM deployment is the more used approach, the adoption of containers has been growing exponentially during the last years [6]. This is due to the deployment and scaling is significantly faster and easier to perform. Furthermore, they require less hardware and software resources, enabling dynamic and lightweight orchestration.

VM-based VNF is a virtual machine with hardware resources, based on an image that hosts the functionality or software. The typical approach when deploying this kind of VNF is to prepare a base image with generic tools and configure it after the deployment using different methods such as Ansible [7]. Another option is to use OSM Day-0 and Day-1 configuration to inject configuration parameters in the VNF. On the other hand, CNFs are another approach to build cloud-native applications. They are composed of microservices, often as open-source and they are hosted in containers, typically using Docker. Besides, they can be leveraged using Container as a Service solutions such as Kubernetes, which have many capabilities in detecting failures and performance degradations and reacting by deploying a new container.

### 3. NetApps initial design report

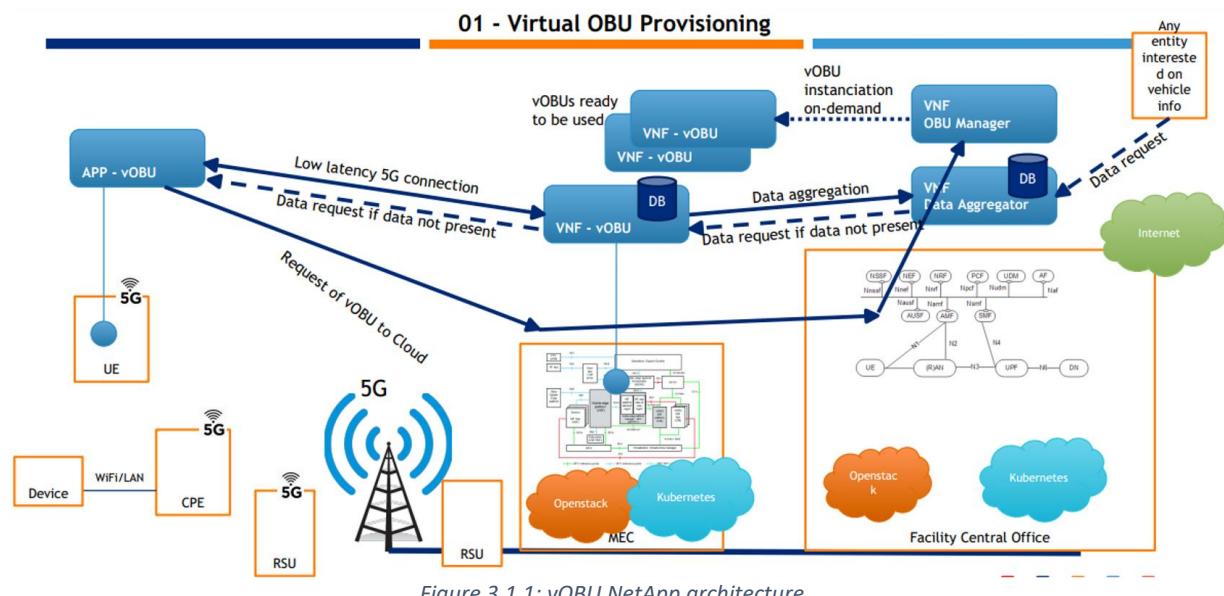
#### 3.1. NetApp #1

##### 3.1.1 NetApp functional overview, current status and next steps

OdinS introduces a solution that provides the necessary vOBUs that are instantiated at the edge of the access network with the purpose of offloading from the physical OBUs computationally intensive tasks to the network, following the MEC approach.

The NetApp is in its first beta version and has been designed, developed, and successfully onboarded locally. It has been deployed in OSM 8 and 10 using a local instance of Openslice, in an effort to simulate the scenario that will be used in the 5GASP framework. By doing this, it has been demonstrated that the NetApp operates correctly in an architecture with the same principles as the one presented in the project. Besides, the NetApp functionality has been evaluated in a local vehicular network to check its capabilities, operating as expected.

Currently, minor changes in the code are being performed in order to optimize its functioning. Furthermore, OdinS is developing a couple of custom test VNFs to perform functional testing on the NetApp. As future steps, the test plan of the NetApp is being designed following the guidelines provided in WP5 [3] and, apart from that, OdinS is also exploring how to adapt the application to containers, instead of VMs, to deploy it using Kubernetes.



As aforementioned, the NetApp has been successfully deployed locally using the capabilities offered by the Murcia testbed. In the following images, the deployment details can be seen. In first place, Fig. 3.1.2 shows the vOBU NSD instantiated from our local OSM10, also, in Fig. 3.1.3 the VNFs are shown. More details can be seen in Fig. 3.1.4, where the Openstack dashboard shows the instantiated VMs together with some network information.

## NS Instances

Init Running / Configured Failed Scaling					
Name	Identifier	Nsd name	Operational Status	Config Status	
surrogatesNetapp	5d501e64-92c6-4c7c-8077-f0f290504ae5	surrogates_OSM10_nsd	✓	✓	

Figure 3.1.2: vOBU NSD instantiated from OSM10

## VNF Instances

Identifier	VNFD	Member Index
Identifier	VNFD	Member Index
23459dcbb-3662-4eb8-9187-f51bbfa8e4fc	management_vnfd	1
b5046b25-82ab-49f9-bd58-4a81b1177fc8	vOBU_vnfd	2
0772da89-be64-4297-93f5-3c161a0ab970	vOBU_vnfd	3
f600d65c-55ce-48ce-a358-fd8bc026e304	vOBU_vnfd	4
b2e8d1fc-a3ec-4fc3-b5ae-dfe098da09e5	aggregator_vnfd	5
2580f5a9-60a4-43a6-bfff-7fd21807b4f	simulatedObu_vnfd	6

Figure 3.1.3: vOBU VNFs instances in OSM10 dashboard

□ surrogatesNetapp-6-simulatedObu-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.102 557-5GASPdata-10G 10.205.57.56
□ surrogatesNetapp-5-aggregator-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.112 557-5GASPdata-10G 10.205.57.76
□ surrogatesNetapp-4-vOBU-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.125 557-5GASPdata-10G 10.205.57.54
□ surrogatesNetapp-3-vOBU-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.101 557-5GASPdata-10G 10.205.57.53
□ surrogatesNetapp-2-vOBU-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.119 557-5GASPdata-10G 10.205.57.78
□ surrogatesNetapp-1-management-vnf-VM-0	debian-baseSurrogates-certs	725-5GASPmgmt-10G 10.207.25.113 557-5GASPdata-10G 10.205.57.57

Figure 3.1.4: vOBU instances as seen in the local Openstack

Once instantiated, the operation of the NetApp can be seen in Figure 3.1.5 and in Figure 3.1.6. In the first one, a simulated physical OBU deployed to emulate a car is requesting a virtual

counterpart using as an identifier its plate number. In the second one, the vOBU manager is waiting for requests and when it receives the request from the simulated OBU, it starts a new vOBU that is attached to the physical one.

```
mac@surrogatesnetapp-6-simulatedobu-vnf-vm-0:~$ python3.8 obu.py 10.207.25.113 0250DZX
Requesting vOBU...
[ '/', '/vehicle', '/vehicle/rpm' ]
CoAP Server: Start listening...
status ERROR
ERROR
status ERROR
ERROR
status ERROR
ERROR
status ERROR
ERROR
status OK
vobu_ip vobu
vobu_id 1
```

Figure 3.1.5: Simulated physical OBU requesting for a vOBU

```
rac@surrogatesnetapp-1-management-vnf-vm-0:~$ python3.8 manager.py
Reading configuration from default config file in [./manager.conf]...
Setting logging level to [DEBUG], logpath [./output_manager.log], append [yes]...
vOBU list:

Main Thread: Starting REST server (TCP) at address: [0.0.0.0] port [8070]car2vobu list:

Starting HTTP server at port 8070
10.207.25.102 - - [25/Jan/2022 11:42:37] "GET /getVobu?plate=0250DZX HTTP/1.1" 200 -
10.207.25.102 - - [25/Jan/2022 11:42:39] "GET /getVobu?plate=0250DZX HTTP/1.1" 200 -
vOBU list:
car2vobu list:
10.207.25.102 - - [25/Jan/2022 11:42:41] "GET /getVobu?plate=0250DZX HTTP/1.1" 200 -
10.207.25.102 - - [25/Jan/2022 11:42:43] "GET /getVobu?plate=0250DZX HTTP/1.1" 200 -
vOBU list:
car2vobu list:
10.207.25.119 - - [25/Jan/2022 11:42:45] "GET /registerVobu?ip=vobu HTTP/1.1" 200 -
10.207.25.102 - - [25/Jan/2022 11:42:45] "GET /getVobu?plate=0250DZX HTTP/1.1" 200 -
vOBU list:
Key: [1] Value: [1/vobu/0250DZX/BUSY]

car2vobu list:
Key: [0250DZX] Value: [1]
vOBU list:
Key: [1] Value: [1/vobu/0250DZX/BUSY]
```

Figure 3.1.6: vOBU manager responding to a vOBU request and assigning it to the simulated OBU

### 3.1.2 NetApp analysis

In this subsection, the NSDs and VNFDs of the NetApp are presented from the point of view of the analysis. Note that the simulated OBU VNF is omitted in this subsection and in the successive ones, as it is used for development purposes, and it is not a part of the NetApp.

#### 3.1.2.1 NSDs

vOBU NSD	
Number of Services (NSDs)	1
Number of total VNFs	3
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information

How is it operated	Configuration of Mobile device requires human intervention. Right now, configuration is done with Ansible scripts and the adaptation to fully automated deployment planned.
Dependencies	No

Table 3.1.1: vOBU NSD

### 3.1.2.2 VNFs

VNF OBU Manager (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM8-10)
Placement (latency from-to)	400-500ms
Internet access	No
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.1.2: VNF OBU Manager

VNF Aggregator (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM8-10)
Placement (latency from-to)	400-500ms
Internet access	Yes
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.1.3: VNF Aggregator

VNF vOBU (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM8-10)
Placement (latency from-to)	400-500ms
Internet access	No
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.1.4: VNF vOBU

### 3.1.2.3 NSD and VNF relations/dependencies

The relationships among the different VNFs of the NetApp can be clearly seen in Figure 3.1.1. The NetApp consists in a single NS that encompasses the three VNFs. The vOBU VNF communicates with the Aggregator to handle the requests and the vOBU Manager is in charge of orchestrating the vOBUs. In this way, the vOBU VNF is dependent on the vOBU Manager and the Aggregator.

### 3.1.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - NSD, 3 VM Roles (VNFD): Manager, Aggregator and the vOBUs themselves plus the physical OBU software (python-based and containerized).
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: No.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires dynamic VM deployment and MANO connectivity to recover deployment status information.
- How is it operated (and does it require manual interventions):
  - Configuration of Mobile device requires human intervention. Right now, configuration with Ansible scripts, adaptation to fully automated deployment planned. Protocol used: IPv6 (CoAP).
- Dependencies (does it expose or consume services from/to other NetApps): No.

vOBU NETAPP'S NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	-
Uplink maximum throughput per UE	-
Isolation level	Virtual resources isolation
V2X Communication mode	Yes, New Radio (NR)
Slice quality of service parameters: 3GPP 5QI	9
Maximum Packet Loss Rate	1%
Supported Device Velocity	3: Vehicular: 10 km/h to 120 km/h

Table 3.1.5: vOBU NetApp's NEST

#### 3.1.3.1 NSDs definitions

```
nsd:  
  nsd:  
    - id: surrogates_OSM10_nsd  
      name: surrogates_OSM10_nsd  
      description: New version of Surrogates Descriptors with OSM10 for test scenario.  
      designer: ODINS  
      version: '1.0'  
      df:
```

```

- id: default-df
vnf-profile:
- id: '1'
  vnf-id: management_vnfd
  virtual-link-connectivity:
    - constituent-cpd-id:
      - constituent-base-element-id: '1'
        constituent-cpd-id: mgmt-vnf-eth0-ext
        virtual-link-profile-id: 725-5GASPmgmt-10G
    - constituent-cpd-id:
      - constituent-base-element-id: '2'
        constituent-cpd-id: mgmt-vnf-eth1-ext
        virtual-link-profile-id: 557-5GASPdata-10G
- id: '2'
  vnf-id: vOBU_vnfd
  virtual-link-connectivity:
    - constituent-cpd-id:
      - constituent-base-element-id: '1'
        constituent-cpd-id: vobu-vnf-eth0-ext
        virtual-link-profile-id: 725-5GASPmgmt-10G
    - constituent-cpd-id:
      - constituent-base-element-id: '2'
        constituent-cpd-id: vobu-vnf-eth1-ext
        virtual-link-profile-id: 557-5GASPdata-10G
- id: '3'
  vnf-id: aggregator_vnfd
  virtual-link-connectivity:
    - constituent-cpd-id:
      - constituent-base-element-id: '1'
        constituent-cpd-id: agg-vnf-eth0-ext
        virtual-link-profile-id: 725-5GASPmgmt-10G
    - constituent-cpd-id:
      - constituent-base-element-id: '2'
        constituent-cpd-id: agg-vnf-eth1-ext
        virtual-link-profile-id: 557-5GASPdata-10G
  virtual-link-desc:
- id: 725-5GASPmgmt-10G
  mgmt-network: 'true'
  vim-network-name: 725-5GASPmgmt-10G
- id: 557-5GASPdata-10G
  vim-network-name: 557-5GASPdata-10G
vnfd-id:
- management_vnfd
- vOBU_vnfd
- aggregator_vnfd
- simulatedObu_vnfd

```

### 3.1.3.2 VNFs definitions

```

vnfd:
description: vOBU entity
id: vOBU_vnfd
product-name: vOBU_vnfd
provider: ODINS
version: '1.0'
df:
- id: default-df
instantiation-level:
- id: default-instantiation-level
vdu-level:
- number-of-instances: 1
  vdu-id: vOBU-vnf-VM
vdu-profile:
- id: vOBU-vnf-VM
  min-number-of-instances: 1

```

```

max-number-of-instances: 4
ext-cpd:
- id: vobu-vnf-eth0-ext
int-cpd:
  cpd: eth0-int
  vdu-id: vOBU-vnf-VM
- id: vobu-vnf-eth1-ext
int-cpd:
  cpd: eth1-int
  vdu-id: vOBU-vnf-VM
mgmt-cp: vobu-vnf-eth0-ext
sw-image-desc:
- id: debian-baseSurrogates-certs
  image: debian-baseSurrogates-certs
  name: debian-baseSurrogates-certs
vdu:
- cloud-init-file: cloud-init.cfg
  description: vOBU-vnf-VM
  id: vOBU-vnf-VM
  name: vOBU-vnf-VM
  sw-image-desc: debian-baseSurrogates-certs
  int-cpd:
    - id: eth0-int
      virtual-network-interface-requirement:
        - name: eth0
          virtual-interface:
            bandwidth: 0
            type: VIRTIO
    - id: eth1-int
      virtual-network-interface-requirement:
        - name: eth1
          virtual-interface:
            bandwidth: 0
            type: VIRTIO
    virtual-compute-desc: vOBU-vnf-VM-compute
    virtual-storage-desc:
      - vOBU-vnf-VM-storage
    virtual-compute-desc:
      - id: vOBU-vnf-VM-compute
        virtual-cpu:
          num-virtual-cpu: 2
        virtual-memory:
          size: 2.0
    virtual-storage-desc:
      - id: vOBU-vnf-VM-storage
        size-of-storage: 50

```

```

vnfd:
description: Management entity
id: management_vnfd
product-name: management_vnfd
provider: ODINS
version: '1.0'
df:
- id: default-df
instantiation-level:
- id: default-instantiation-level
  vdu-level:
    - number-of-instances: 1
      vdu-id: management-vnf-VM
  vdu-profile:
    - id: management-vnf-VM
      min-number-of-instances: 1
ext-cpd:
- id: mgmt-vnf-eth0-ext
int-cpd:

```

```

cpd: eth0-int
vdu-id: management-vnf-VM
- id: mgmt-vnf-eth1-ext
int-cpd:
cpd: eth1-int
vdu-id: management-vnf-VM
mgmt-cp: mgmt-vnf-eth0-ext
sw-image-desc:
- id: debian-baseSurrogates-certs
image: debian-baseSurrogates-certs
name: debian-baseSurrogates-certs
vdu:
- cloud-init-file: cloud-init.cfg
description: management-vnf-VM
id: management-vnf-VM
name: management-vnf-VM
sw-image-desc: debian-baseSurrogates-certs
int-cpd:
- id: eth0-int
virtual-network-interface-requirement:
- name: eth0
virtual-interface:
bandwidth: 0
type: VIRTIO
- id: eth1-int
virtual-network-interface-requirement:
- name: eth1
virtual-interface:
bandwidth: 0
type: VIRTIO
virtual-compute-desc: management-vnf-VM-compute
virtual-storage-desc:
- management-vnf-VM-storage
virtual-compute-desc:
- id: management-vnf-VM-compute
virtual-cpu:
num-virtual-cpu: 2
virtual-memory:
size: 2.0
virtual-storage-desc:
- id: management-vnf-VM-storage
size-of-storage: 50

```

```

vnfd:
description: Aggregator entity
id: aggregator_vnfd
product-name: aggregator_vnfd
provider: ODINS
version: '1.0'
df:
- id: default-df
instantiation-level:
- id: default-instantiation-level
vdu-level:
- number-of-instances: 1
vdu-id: aggregator-vnf-VM
vdu-profile:
- id: aggregator-vnf-VM
min-number-of-instances: 1
ext-cpd:
- id: agg-vnf-eth0-ext
int-cpd:
cpd: eth0-int
vdu-id: aggregator-vnf-VM
- id: agg-vnf-eth1-ext
int-cpd:

```

```

cpd: eth1-int
vdu-id: aggregator-vnf-VM
mgmt-cp: agg-vnf-eth0-ext
sw-image-desc:
- id: debian-baseSurrogates-certs
  image: debian-baseSurrogates-certs
  name: debian-baseSurrogates-certs
vdu:
- cloud-init-file: cloud-init.cfg
  description: aggregator-vnf-VM
  id: aggregator-vnf-VM
  name: aggregator-vnf-VM
  sw-image-desc: debian-baseSurrogates-certs
int-cpd:
- id: eth0-int
  virtual-network-interface-requirement:
    - name: eth0
      virtual-interface:
        bandwidth: 0
        type: VIRTIO
- id: eth1-int
  virtual-network-interface-requirement:
    - name: eth1
      virtual-interface:
        bandwidth: 0
        type: VIRTIO
  virtual-compute-desc: aggregator-vnf-VM-compute
  virtual-storage-desc:
    - aggregator-vnf-VM-storage
  virtual-compute-desc:
- id: aggregator-vnf-VM-compute
  virtual-cpu:
    num-virtual-cpu: 2
  virtual-memory:
    size: 2.0
  virtual-storage-desc:
    - id: aggregator-vnf-VM-storage
      size-of-storage: 50

```

### 3.1.4 NetApp's tests plan and definition

The NetApp will follow the guidelines provided by WP5 [3] for the test plan. In WP5, the initial testplan is led by the NetApp definitions that were established in WP2/D2.1 [8]. In this way, these tests are defined to check if a certain NetApp complies with the NetApp general definition. For the vOBU NetApp, the selected mandatory tests are the ones related to definitions 1, 3, 4 and 19; and the chosen optional tests are related to definitions 2, 6, 8, 13, 15 and 18.

Furthermore, the NetApp also executes performance tests to check that the infrastructure is working properly, and it can support the operation of the vOBUs. These tests check the end-to-end latency, packet delivery ratio and jitter.

Regarding the functional testing of the NetApp, as aforementioned in the work in progress subsection, custom test VNFs are being developed to validate the correct operation of the service itself. Concretely, these VNFs will act as clients of the NetApp, and they will perform a single request to evaluate the correctness of the answers and a batch of requests to check the performance of the service.

The test descriptor will be developed to include all these tests by defining the different test cases and their parameters, as well as the execution order and results' collection. The tests will be implemented in Python scripts, which will feed the Robot framework. The custom test VNFs will be prepared based on Debian images.

To allow the testing of the vOBU NetApp there are two main hardware/physical conditions. The first one is to evaluate the operation of the NetApp with a static physical OBU and the second one is with a moving physical OBU (onboarded in a car). Regarding the network conditions, the NetApp can be tested in isolation (no external traffic), or in a shared medium with heavy traffic that affects the network performance.

All the testing aspects discussed will be collected in the test plan and in the test descriptor. The final outcome of the testing phase will be given by the successful passing of all the aforementioned test cases.

### 3.1.5 NetApp requirements for testbeds

The vOBU NetApp requirement for the testbed is the physical OBU, which must be present in the testbeds capable of hosting the NetApp. This physical OBU does not need high-end capabilities, it can be similar to a Raspberry Pi 4, and it must include a 5G modem to be able to connect to the network. Optionally, although desired, it should include an 802.11p [9] modem to experiment with multi-RAT access. The OBU client software is made in Python, and it will be provided to the testbeds in the form of a Docker container ready to deploy. Finally, the physical OBU should be connected to sensors of a vehicle to gather automotive info for the NetApp functioning. However, as an initial approach, this information can be substituted by dummy data produced by the client software.

## 3.2. NetApp #2

### 3.2.1 NetApps functional overview, current status and next steps

In the context of the 5GASP project, NetApp #2 is the virtual representation of the Road-side Intelligent Transport System Station (R-ITS-S), or what is commonly agreed to be named Virtual Road-side-Unit (vRSU) and part of the automotive verticals NetApps. Following the MEC approach, the edge instantiation-based NetApp intends to offload the computational process from the physical R-ITS-S and ensure low latencies responses due to the proximity of the computing facilities to the point of attachment considering the time-critical Cooperative ITS use-cases. NetApp #2 identified functionalities:

- Support communication over short-range communication interfaces (802.11p [9], 5G D2D, C-V2X)
- Optimize data exchange between the Edge and Core NetApps
- Trigger Cooperative ITS use-cases covering a local area (traffic light phases, Hazard Notifications, ...)
- Trigger Cooperative ITS use-cases that require short-range and/or time-critical, e.g., Platooning, Collision Warning, ...

- Reduce and optimize communication latency with V-ITS-S

The NetApp is in the development phase, and the definition of VNF and NSD are based on the ongoing tests of a Proof-of-Concept (PoC) deployment. Following the design methodology for a Proof-of-Concept (PoC) NetApp in D4.2 [2], NetApp #2 has finalized PHASE-I and PHASE-II regarding the analysis, services definition, and functionalities identification. However, the deployment of PoC NetApp is ongoing at the Murcia site. In addition, the refinement of both VNFs and NSDs are also continuous alongside the development advancing to plan the final deployment of YoGoKo two NetApps at the Murcia site.

Figure 3.2.1 depicts the general architecture overview, the deployment, and exchange scenarios of the two automotive NetApps (NetApp #2 vRSU NetApp and NetApp #3 C-ITS-S NetApp).

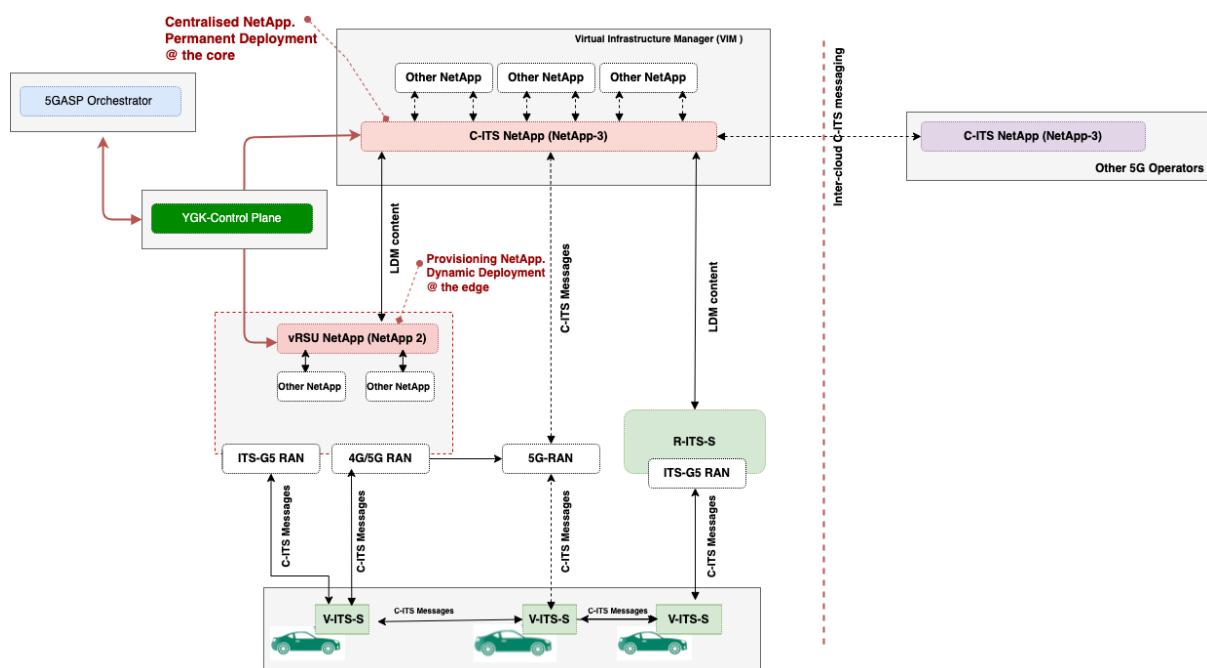


Figure 3.2.1: vRSU NetApp (NetApp #2) and C-ITS-S NetApp (NetApp #3) general architecture

### 3.2.2 NetApp analysis

The data exchange between automotive NetApps, UEs, and the ITS stations must comply with C-ITS standards respecting the standardized data formats, communication protocols, communication paths, and security requirements. Therefore, the data flow of vRSU (NetApp #2) and the C-ITS-S NetApps (NetApp #3) were key analysis aspects to identify the exchange between different NetApps considering various interfaces.

#### Data Flow Analysis

YoGoKo has chosen the dataflow-based NetApp analysis to identify the exchange between NetApps to recognize the functionalities and the proper tests when the automotive-like NetApps deployment occurs. Such extended analysis is due to the fact that data format

mismatch is expected to be an issue, especially for direct communication through the ShortRange interface (such as ITS-G5 [10]).

The dataflow analysis evaluates four primary data sources, considering the exchange between NetApp #2 and NetApp #3 as depicted in Figure 3.2.1 (therefore, the flow analysis of NetApp #3 in the subsequent sections will be referred to the common analysis presented for NetApp #2):

- External NetApp or any Third-Party Cloud
- NetApp #3 (Cooperative C-ITS-S NetApp)
- NetApp #2 (Virtual RSU NetApp)
- Vehicle ITS Station (V-ITS-S)

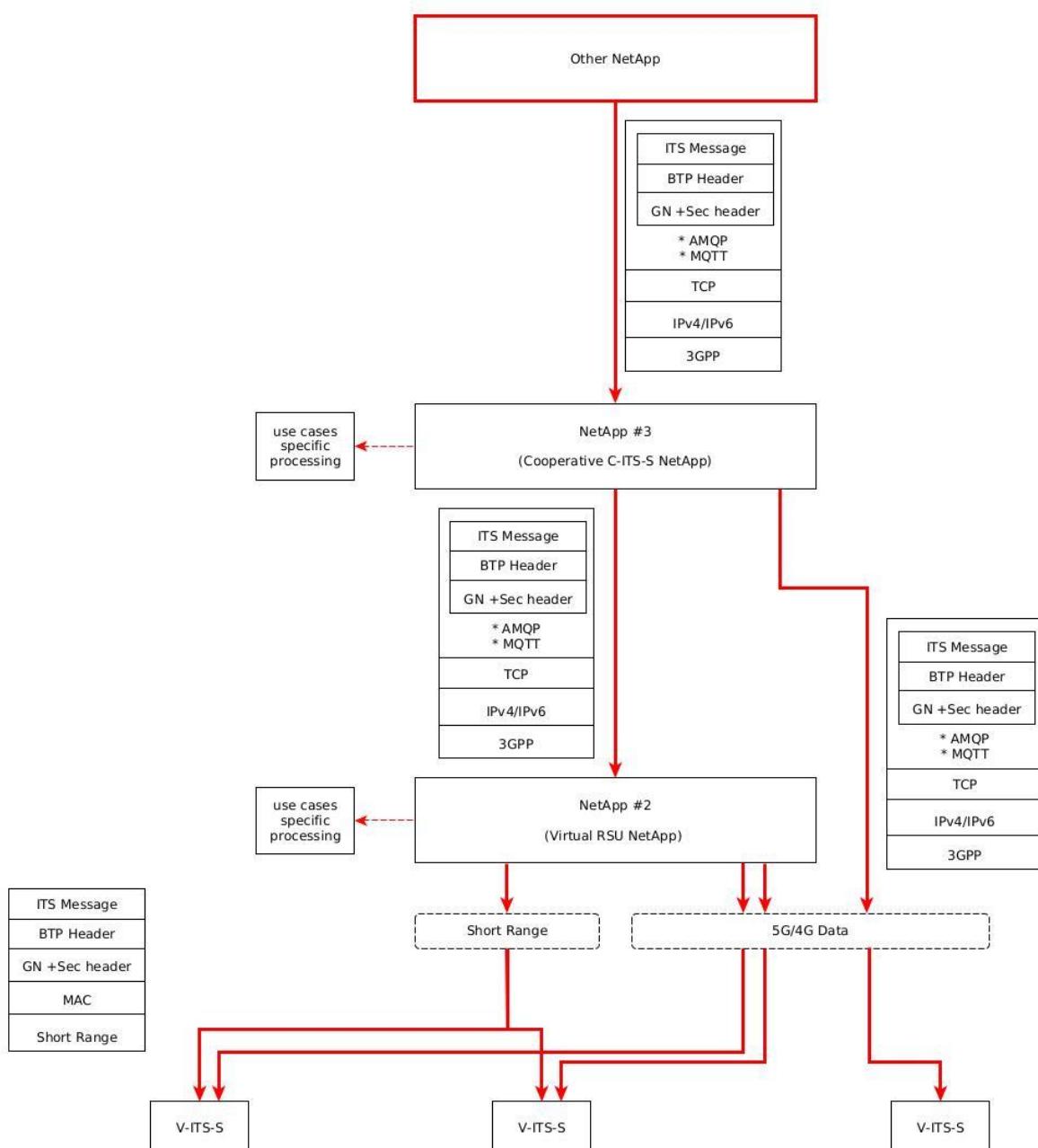


Figure 3.2.2: Data flows with External NetApp as the data source

### Data flows with External NetApp as the data Source

Scheme in Figure 3.2.2 represents the possible data flows when any other Automotive-like NetApp performs an exchange (e.g., trigger an ITS message):

- The exchange must go through NetApp #3 (C-ITS NetApp), and all ITS messages triggered by other external NetApps must be transmitted to the core-based NetApp instance.
- Once received by NetApp #3, two different paths are to be considered.
  - The ITS message is then forwarded to the edge-based NetApp #2 (vRSU) instance in order to deliver the message of interest to the Vehicle ITS Station.
  - Otherwise, the ITS message can be directly forwarded to Vehicle ITS Stations.
- Considering the reach of V-ITS-S through NetApp #2, the edge deployed NetApp will then be in charge of forwarding ITS messages to Vehicle ITS Station through Short Range (ITS-G5 [10]) or 4G/5G interfaces. Moreover, NetApp #2 must have the capability to identify localized reachability. If for example the mobile ITS-S is outreached the coverage range limitation of the edge-deployed NetApp #2, a handover is required to assure that message is forwarded, either over ITS-G5 [10] or 4G/5G.

### Data flows with NetApp #3 local processing module as the data Source

For an ITS Message triggered by a specific local processing module, NetApp #3 data flows are slightly different. Once the ITS message has reached NetApp #3, it will have the same downstream flow as the previous example. Meanwhile, it will reach any connected External Netapps in the upstream flow, as depicted in Figure 3.2.3.

When the local processing associated with NetApp #3 triggers an ITS message, the ITS message should be transmitted to the NetApp #3 instance:

- Upstream flow:
  - NetApp #3 will then transmit the ITS Message to any connected external NetApp compatible and aware of such content.
- Downstream flow
  - Once received by NetApp #3, the same two different paths as the previous example (External NetApp as data source) are to be considered.

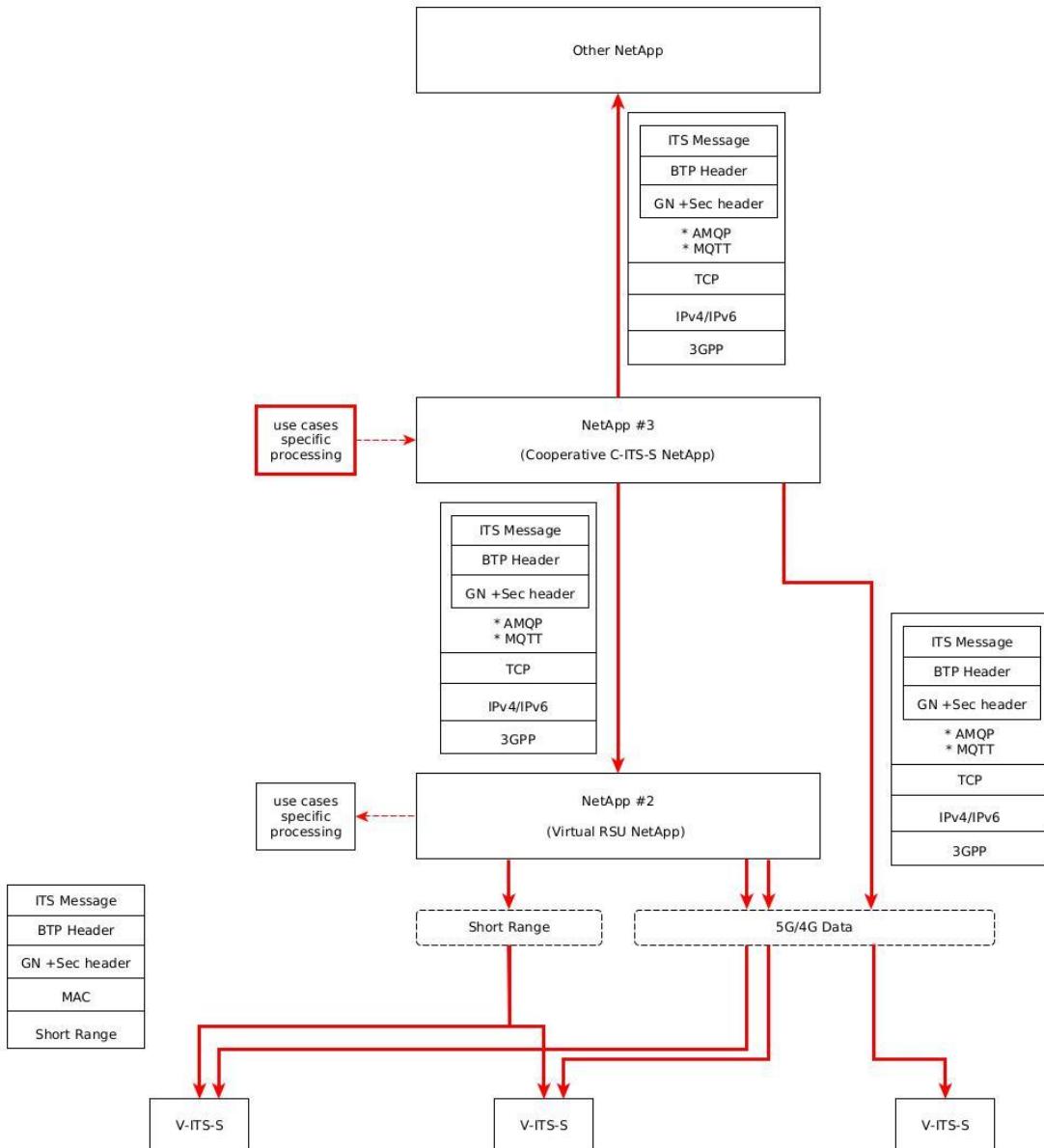


Figure 3.2.3: Data flows between NetApp #3 local processing module as the data Source

#### Data flows with NetApp #2 local processing module as the data Source

In a similar way, the following scheme in Figure 3.2.4 represents the possible data flows when an ITS message is triggered by the local processing module of NetApp #2. The triggering of the ITS message should either be based on the result of relevant ITS information related to the NetApp #2 local area or directly triggered by a third-party application connected to the NetApp instance.

When the local processing triggers an ITS Message, the edge deployed NetApp #2 will manage such and must be received by the NetApp #2 instance:

- Upstream flow
  - For the upstream depicted in the following figure, NetApp #2 should transmit the ITS message to NetApp #3, so that the ITS message reaches any connected External NetApp listening to this type of message.

- Downstream flow
  - In downstream flow, ITS Message should be transmitted to Vehicle ITS Station. This forwarding should be done through ShortRange communication (ITS-G5 [10]) if the distance between the ShortRange communication Interface (IEEE802.11p [9]) of NetApp #2 and Vehicle ITS Station allows it. Otherwise, the ITS message will be forwarded through 4G/5G data.

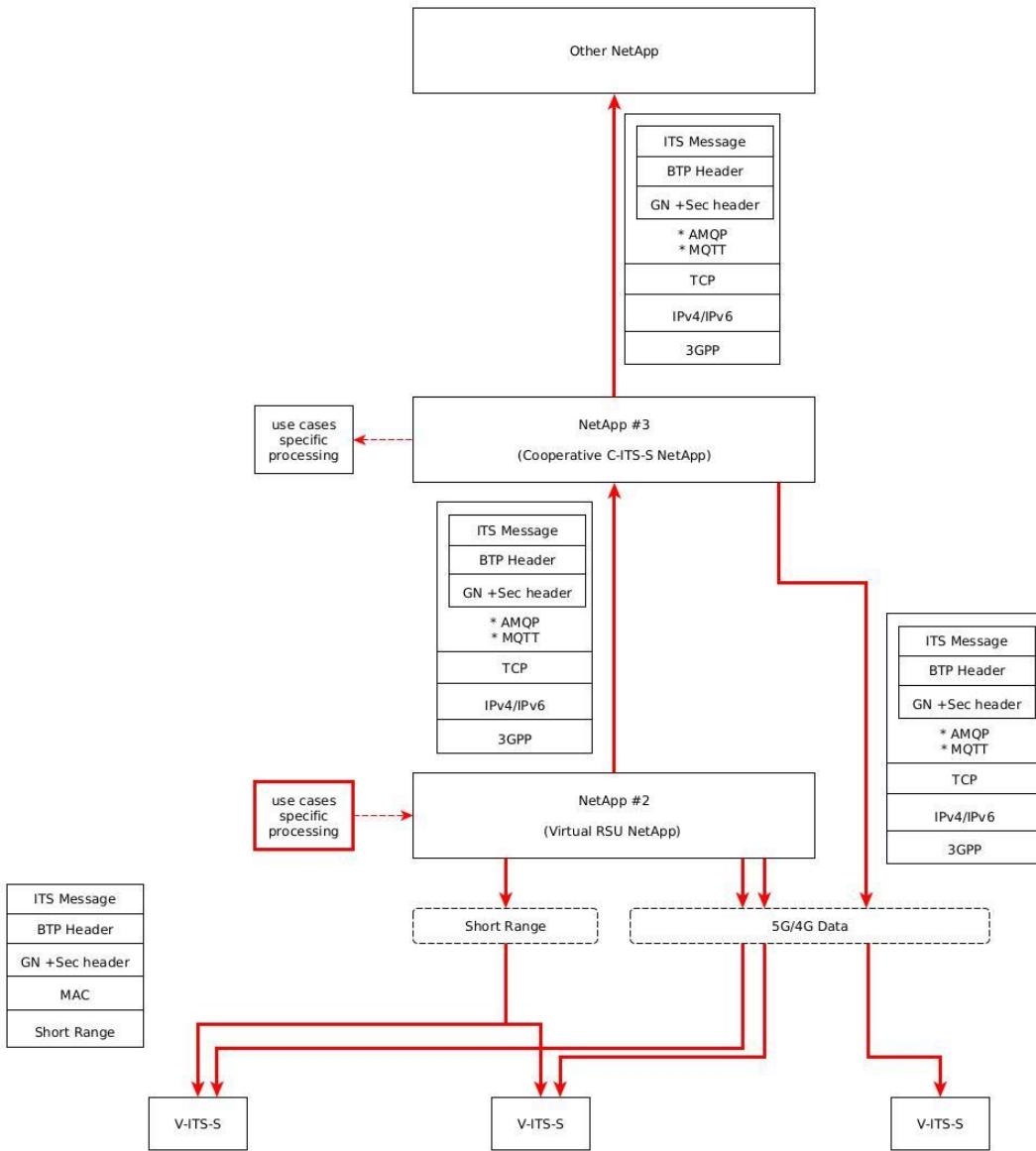


Figure 3.2.4: Data flows NetApp #2 local processing module as the data Source

#### Data flows with Vehicle ITS Station as the data Source

When it comes to the physical Vehicle ITS station (V-ITS-S) as a data source and triggering an ITS message, three sub-schemes are dedicated to identifying the scenarios and associated flows: Vehicle to NetApp #3 instance (Figure 3.2.5), Vehicle to NetApp #2 through Short Range communication protocol ITS-G5 (Figure 3.2.6), and Vehicle to NetApp #2 through 4G/5G Data communication protocol (Figure 3.2.7).

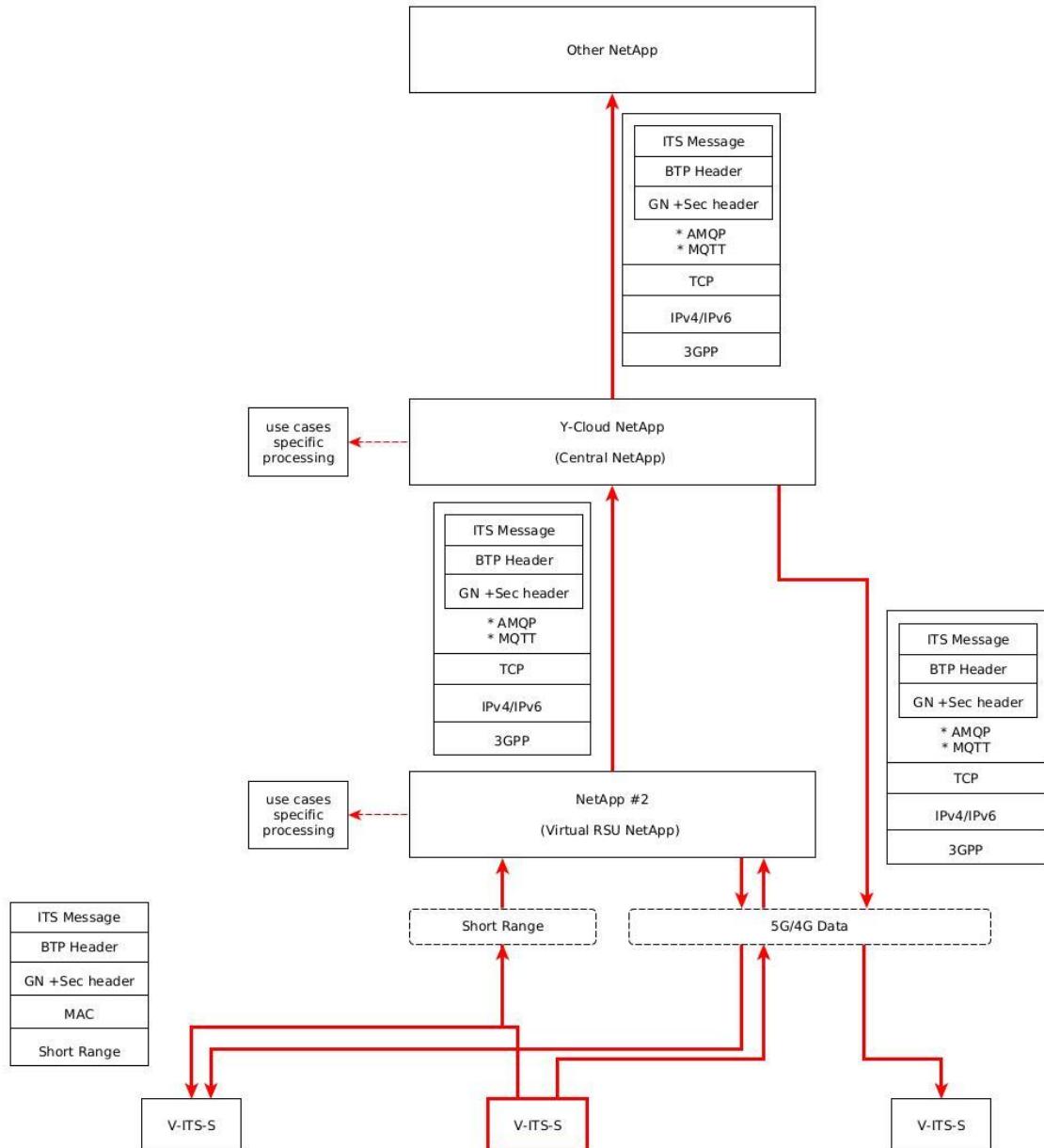


Figure 3.2.5: Data flows V-ITS-S as the data Source

- Vehicle to NetApp #3 (Figure 3.2.5 illustrates the case when an ITS station is the data source and the list of flows to reach NetApp #3)
  - When a Vehicle ITS Station triggers an ITS Message through 4G/5G data communication interface, the message is forwarded to NetApp #3 directly:
    - Upstream flow
      - Once received by NetApp #3 the ITS message should be forwarded to both NetApp #3 local processing and any connected external NetApp listening to this type of message.
    - Downstream flow
      - Once received by NetApp #3, the ITS message should be forwarded to NetApp #2.

- Then NetApp #2 will transmit ITS message to both NetApp #3 local processing and Vehicle ITS Station either through Short Range communication (ITS-G5) or 4G/5G data communication interface.
- Vehicle to NetApp #2 through Short Range (reaching the vehicle ITS station over the localized short-range communication and using the ITS-G5 is the second case depicted in the Figure 3.2.6)

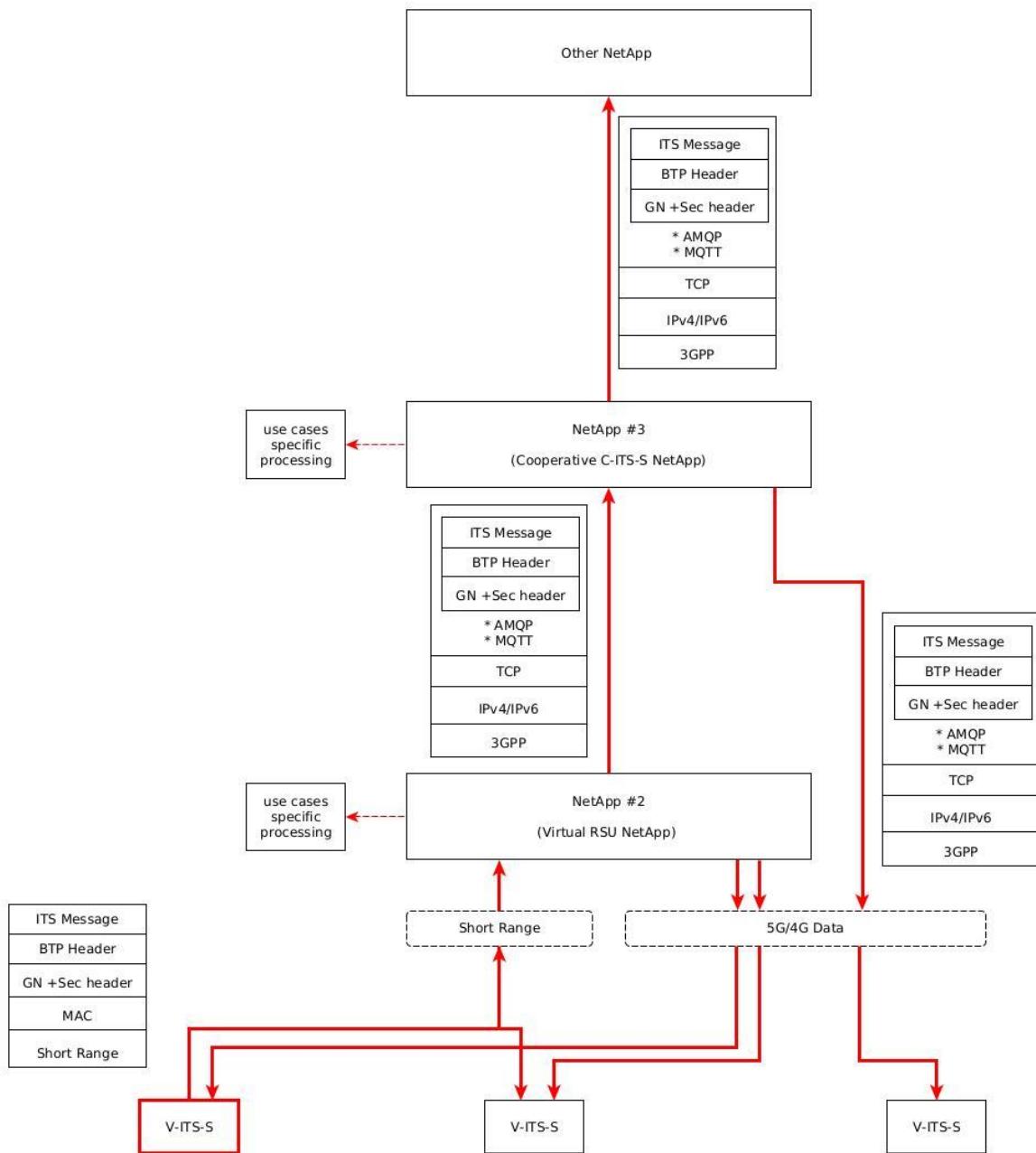


Figure 3.2.6: Data flows V-ITS-S to NetApp #2 through short-range interfaces

- When a Vehicle ITS Station triggers an ITS Message through Short Range communication interface, the message is forwarded to both Vehicle ITS Station and NetApp #2 in the sensing range.

- Upstream flow
  - Once received by NetApp #2, the ITS message will be forwarded to both NetApp #2 local processing and NetApp #3.
  - Finally, ITS message should be forwarded from NetApp #3 to any connected external NetApp.
- Downstream flow
  - Once received by NetApp #2, the ITS message can directly be forwarded to Vehicle ITS Station through 4G/5G data.
- Vehicle to NetApp #2 through 4G/5G (Figure 3.2.7 illustrates the use of 4G/5G connectivity to reach the vehicle ITS station)

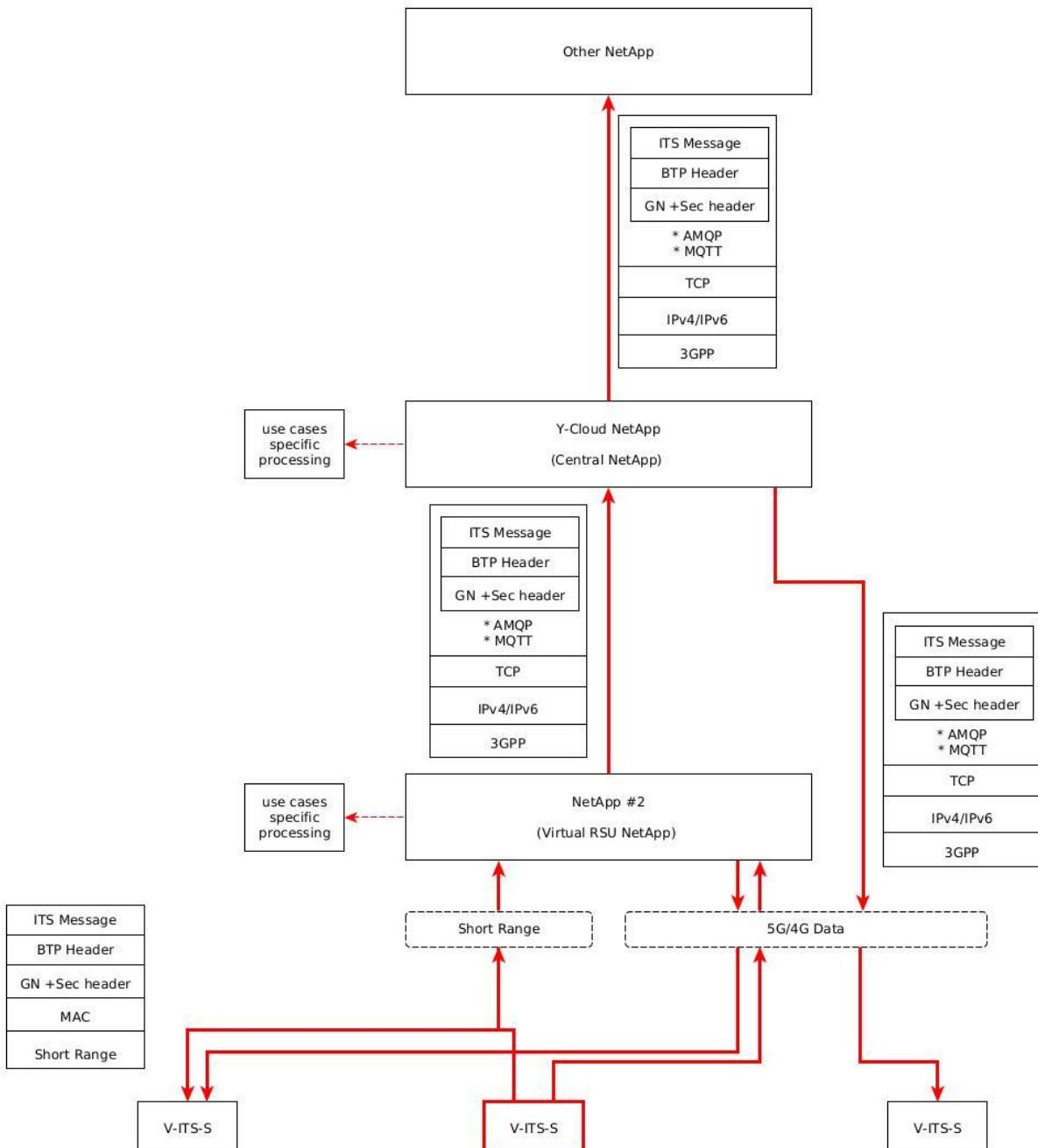


Figure 3.2.7: Data flows V-ITS-S to NetApp #2 through 4G/5G interfaces

- When a Vehicle ITS Station triggers an ITS Message through the 4G/5G data communication interface, the message is forwarded to NetApp #2.
  - Upstream flow
    - Once received by NetApp #2, the ITS message will be forwarded to both NetApp #2 local processing and NetApp #3
    - ITS message is then will be forwarded from NetApp #3 to any connected external NetApp and local processing module.
  - Downstream flow
    - Once received by NetApp #2, the ITS message can directly be forwarded to Vehicle ITS Station through 4G/5G data.

### 3.2.2.1 NSDs

vRSU NSD	
Number of Services (NSDs)	1
Number of total VNFs	2
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information
How is it operated	Configuration is to be done manually for the moment
Dependencies	No

Table 3.2.1: vRSU NSD

### 3.2.2.2 VNFs

VNF 5GASP Orchestrator (NSD1)	
Number of total VNFs	2
Packaging Info:	VNF (OSM10)
Placement (latency from-to)	400-500ms
Internet access	Yes
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.2.2: vRSU manager VNF

VNF vRSU (NSD1)	
Number of total VNFs	2
Packaging Info:	VNF (OSM8-10)
Placement (latency from-to)	400-500ms
Internet access	Yes
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.2.3: vRSU VNF

### 3.2.2.3 NSD and VNF relations/dependencies

The relationships among the different VNFs of the NetApp can be seen clearly in Figure 3.2.1. The NetApp #2 (vRSU) consists in a single NS that encompasses 2 VNFs. The vRSU VNF communicates with the 5GASP Orchestrator to ensure the orchestration of the vRSU instances. In this way, the vRSU VNF is dependent on the 5GASP Orchestrator. Once running the different vRSU instances, each covering a specific small area will communicate with V-ITS-S and NetApp #2.

The vRSU PoC NetApp has a single NS with a template-based two VNF descriptor as a basic VNF definition used to define fundamental functionalities for onboarding validation as depicted in Figure 3.2.1. The vRSU VNF communicates with the 5GASP Orchestrator to ensure the orchestration of the vRSU instances. Once running the different vRSU instances, each covering a specific small area will communicate with V-ITS-S and NetApp #2. However, more specific VNFs are in the development phase to support the vRSU orchestration and aggregations. Therefore, the vRSU VNF will depend on the 5GASP Orchestrator and management VNFs that are parts of the next development phases.

## 3.2.3 NetApp definition

NetApp #2 (vRSU) definition:

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Virtual Machine
  - Common computing requirements (1vCPU, 1GB RAM, 10GB Storage)
  - Three network interfaces supporting IP(v4/v6) communications
    - Downlink interface to communicate with UEs or to connect with a chained NFV on the downlink path
    - Uplink interface to connect to the Central C-ITS station, and/or other NetApps/NFVs which implement C-ITS use-cases
    - Management interface to allow the control of the NetApp
  - May use direct access to some radio interfaces (802.11p [9], C-V2X)
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs:
  - Simple Network Management Protocol (SNMP) metrics provided
  - Control and configuration through YoGoKo API
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires one or several of the following radio capabilities: 5G/4G/3G data (Uplink/Downlink), PC-5/C-V2X (Sidelink), ITS-G5 [10]/WAVE/802.11p [9], 5G NR-V2X/D2D
  - May need slicing and specific QoS to perform low latency and time critical communications based on 5G/4G/3G data
  - Orchestrated dynamic VM deployment and MANO connectivity to recover deployment status information
- How is it operated (and does it require manual interventions):
  - Each vRSU deployed on the Edge covers a limited geographic area which corresponds to the theoretical radio coverage of the underlying eNB station.
  - A vRSU may be operated as an isolated node or connected to a Central C-ITS station.

- The vRSU uses its downlink interface and/or direct access to radio interfaces to communicate with UEs (e.g., vehicles, connected infrastructure or nomadic devices) in the coverage of the eNB.
- The vRSU uses its uplink interface to exchange information to the Central C-ITS station or other NetApps implementing C-ITS use cases.
- The vRSU uses its management interface for control-plane operations such as orchestration or communication of metrics.
- vRSUs are deployed unconfigured and configuration is managed through YoGoKo configuration API.
- The exchange of data between UE and the vRSU must comply with C-ITS standards (data formats, communication protocols, security, etc.).
- Once the infrastructure is in place, any type of service (able to exploit data transmitted between vehicles and the roadside) could be developed either to demonstrate the benefit of virtual RSU for existing C-ITS services or to demonstrate how new C-ITS services can be developed.
- Dependencies (does it expose or consume services from/to other NetApps): Yes, Efficient MEC handover NetApp, vOBU NetApp, vRSU NetApp and Multi-domain Migration NetApp.

vRSU NETAPP'S NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	-
Uplink maximum throughput per UE	-
Isolation level	Virtual resources isolation
V2X Communication mode	Yes, New Radio (NR)
Slice quality of service parameters: 3GPP 5QI	84 (for time-critical road safety services)
Maximum Packet Loss Rate	1%
Supported Device Velocity	Vehicular: 10 km/h to 120 km/h

Table 3.2.4: vRSU NetApp (NetApp #2) NEST

### 3.2.3.1 NSDs definitions

The following file defines NSD for OSM 9 considering the PoC NetApp:

```
nsd:
  nsd:
    - id: surrogates_OSM10_nsd
      name: surrogates_OSM10_nsd
      description: New version of Surrogates Descriptors with OSM10 for test scenario.
      designer: ODINS
      version: '1.0'
      df:
        - id: default-df
          vnf-profile:
            - id: '1'
              vnfd-id: vRSU_vnfd
              virtual-link-connectivity:
```

```

- constituent-cpd-id:
- constituent-base-element-id: '1'
  constituent-cpd-id: vrsu-vnf-eth0-ext
  virtual-link-profile-id: 725-5GASPmgmt-10G
- constituent-cpd-id:
- constituent-base-element-id: '2'
  constituent-cpd-id: vrsu-vnf-eth1-ext
  virtual-link-profile-id: 557-5GASPdata-10G
- id: '2'
vnfd-id: orchestrator_vnfd
virtual-link-connectivity:
- constituent-cpd-id:
- constituent-base-element-id: '1'
  constituent-cpd-id: orc-vnf-eth0-ext
  virtual-link-profile-id: 725-5GASPmgmt-10G
- constituent-cpd-id:
- constituent-base-element-id: '2'
  constituent-cpd-id: orc-vnf-eth1-ext
  virtual-link-profile-id: 557-5GASPdata-10G
virtual-link-desc:
- id: 725-5GASPmgmt-10G
  mgmt-network: 'true'
  vim-network-name: 725-5GASPmgmt-10G
- id: 557-5GASPdata-10G
  vim-network-name: 557-5GASPdata-10G
vnfd-id:
- orchestrator_vnfd
- vRSU_vnfd

```

### 3.2.3.2 VNFs definitions

The following defines the vRSU and orchestrator VNF for OSM 9 considering the vRSU PoC NetApp:

```

vnfd:
  description: vRSU entity
  id: vRSU_vnfd
  product-name: vRSU_vnfd
  provider: YOGOKO
  version: '1.0'
  df:
    - id: default-df
      instantiation-level:
        - id: default-instantiation-level
          vdu-level:
            - number-of-instances: 1
              vdu-id: vRSU-vnf-VM
          vdu-profile:
            - id: vRSU-vnf-VM
              min-number-of-instances: 1
              max-number-of-instances: 4
  ext-cpd:
    - id: vrsu-vnf-eth0-ext
  int-cpd:
    cpd: eth0-int
    vdu-id: vRSU-vnf-VM
  - id: vrsu-vnf-eth1-ext
  int-cpd:
    cpd: eth1-int
    vdu-id: vRSU-vnf-VM
  mgmt-cp: vrsu-vnf-eth0-ext
  sw-image-desc:
    - id: debian-baseSurrogates-certs
      image: debian-baseSurrogates-certs
      name: debian-baseSurrogates-certs
  vdu:

```

```

- cloud-init-file: cloud-init.cfg
description: vRSU-vnf-VM
id: vRSU-vnf-VM
name: vRSU-vnf-VM
sw-image-desc: debian-baseSurrogates-certs
int-cpd:
- id: eth0-int
  virtual-network-interface-requirement:
    - name: eth0
      virtual-interface:
        bandwidth: 0
        type: VIRTIO
- id: eth1-int
  virtual-network-interface-requirement:
    - name: eth1
      virtual-interface:
        bandwidth: 0
        type: VIRTIO
virtual-compute-desc: vRSU-vnf-VM-compute
virtual-storage-desc:
- vRSU-vnf-VM-storage
virtual-compute-desc:
- id: vOBU-vnf-VM-compute
  virtual-cpu:
    num-virtual-cpu: 2
  virtual-memory:
    size: 2.0
  virtual-storage-desc:
- id: vRSU-vnf-VM-storage
  size-of-storage: 50

```

```

vnfd:
description: Orchestrator entity
id: orchestrator_vnfd
product-name: orchestrator_vnfd
provider: YOGOKO
version: '1.0'
df:
- id: default-df
instantiation-level:
- id: default-instantiation-level
  vdu-level:
    - number-of-instances: 1
      vdu-id: orchestrator-vnf-VM
vdu-profile:
- id: orchestrator -vnf-VM
  min-number-of-instances: 1
ext-cpd:
- id: orc-vnf-eth0-ext
int-cpd:
  cpd: eth0-int
  vdu-id: management-vnf-VM
- id: orc-vnf-eth1-ext
int-cpd:
  cpd: eth1-int
  vdu-id: orchestrator -vnf-VM
mgmt-cp: orc-vnf-eth0-ext
sw-image-desc:
- id: debian-baseSurrogates-certs
  image: debian-baseSurrogates-certs
  name: debian-baseSurrogates-certs
vdu:
- cloud-init-file: cloud-init.cfg
  description: orchestrator -vnf-VM
  id: orchestrator-vnf-VM
  name: orchestrator-vnf-VM

```

```

sw-image-desc: debian-baseSurrogates-certs
int-cpd:
- id: eth0-int
  virtual-network-interface-requirement:
  - name: eth0
    virtual-interface:
      bandwidth: 0
      type: VIRTIO
- id: eth1-int
  virtual-network-interface-requirement:
  - name: eth1
    virtual-interface:
      bandwidth: 0
      type: VIRTIO
virtual-compute-desc: orchestrator-vnf-VM-compute
virtual-storage-desc:
- orchestrator-vnf-VM-storage
virtual-compute-desc:
- id: orchestrator-vnf-VM-compute
  virtual-cpu:
    num-virtual-cpu: 2
  virtual-memory:
    size: 2.0
virtual-storage-desc:
- id: orchestrator-vnf-VM-storage
  size-of-storage: 50

```

### 3.2.4 NetApp's tests plan and definition

The ongoing test definitions consider the associated functionalities tests and follow the guidelines and test plane detailed in WP5. Both NetApp will be tested on the mandatory tests definitions 4 and 1. After identifying the test cases in D5.3 [3], a test descriptor will be developed accordingly.

Besides the virtualization and functionalities tests, NetApp #2 is expected to interact with physical OBUs and possibly other RSUs in the sensing range, either over the 802.11p [9] or any other radio access technology.

The vRSU is meant to be deployed on the edge and may directly use radio resources for short-range communications (e.g., 802.11p or C-V2X). Since 802.11p and PC-5/C-V2X mode-4 share the same properties (for layer 3 and above), we may consider focussing on one of these technologies for testing.

Since the vRSU may have several downlink interfaces, and may or may not have communications with other NetApps or a Central C-ITS station, the construction of test cases will need to consider the high combinatory of possible situations (e.g., a C-ITS message coming from a vehicle in C-V2X being retransmitted to the Central C-ITS Station, to other NetApps, and to UEs connected in 5G data at the same time).

Since the vRSU is limited to a specific geographic zone, test cases will also need to bring special attention to what happens when a UEs enters or leaves the zone. This may require dynamic testing (with a real or fake positioning).

Examples of test cases:

- A UE in the zone sends a message over 5G/4G/3G data, this message must be transmitted to other NetApps deployed on the edge, to the Central C-ITS Station, and to other UEs connected over 5G/4G/3G data and 802.11p or C-V2X.
- A UE out of the zone sends a message over 5G/4G/3G data, this message must be ignored.
- The Central C-ITS Station sends a message, this message must be transmitted to UEs over 5G/4G/3G data and 802.11p or C-V2X, and to other NetApps deployed on the edge.

The following table summarizes a set of service basic test examples for the vRSU NetApp (NetApp #2).

ID	Definition	Test point	Expected value	Test method
1	A C-ITS message originated from the UE is received by other nodes	Application Plane	PASS/FAIL	Automated or Manual
2	A C-ITS message originated from the UE out of the coverage zone of the vRSU is not received by other nodes	Application Plane	PASS/FAIL	Automated or Manual
3	A C-ITS message originated from a NetApp deployed on the edge is received by other nodes	Application Plane	PASS/FAIL	Automated or Manual

Table 3.2.5: Service test examples for the C-ITS NetApps (NetApp #2)

### 3.2.5 NetApp requirements for testbeds

Tests for the vRSU may be fully virtualised by using stub code to simulate data coming from radio interfaces, in which case no specific facility is required from testbeds, or it may be semi-virtualised, in which case radio resources will be requested from testbeds. HW requirements:

- eNB with 5G/4G/3G data support, and ITS-G5/802.11p or C-V2X (ideally both)
- UEs with 5G/4G/3G data
- UEs with ITS-G5/802.11p or C-V2X
- UEs with 5G/4G/3G data + ITS-G5/802.11p or C-V2X
- All UEs must be geo-localized with an emulated or real positioning solution and radio coverage should be controlled consistently with positioning
- All UEs must be able to execute a C-ITS software stack (ARM or x86, 1CPU, 1GB RAM, 512MB Storage)

## 3.3. NetApp #3

### 3.3.1 NetApp requirements for testbeds

In relation with the NetApp #2, the core-based deployment of a virtualized Cooperative ITS-S (C-ITS-S) represents NetApp #3. This NetApp contains the minimum ITS station facilities layer services independently of the communication protocols, allowing users to develop new

applications and services for the Automotive and PPDR verticals while ensuring that the developments are compatible with C-ITS standards. Furthermore, NetApp #3 can also operate as a centralized C-ITS station allowing message forwarding and interaction with other core-based NetApps. Our NetApp analysis section presents more details, especially when C-ITS services might require a wide area coverage. NetApp #3 identified functionalities:

- Forward Cooperative ITS messages from and to 5G connected vehicles and infrastructures.
- Interconnect with external servers (other NetApp, public or private road administrator servers).
- Trigger Cooperative ITS use-cases covering a wide area (Service Advertisement, Traffic Condition...)
- Providing Cooperative ITS coverage over a wide area.

The NetApp is in development phase, and the definition of VNF and NSD are based on the ongoing tests of a PoC NetApp deployment. Following the design methodology for a Proof-of-Concept (PoC) NetApp in D4.2 [2], NetApp #3 have finalized PHASE-I and PHASE-II regarding the analysis, services definition, and functionalities identification. However, the deployment of PoC NetApp is ongoing at Murcia site. In addition, the refinement of both VNFs and NSDs are also continuous alongside the development advancing to plan the final deployment of YoGoKo two NetApps at the Murcia site.

### 3.3.2 NetApp analysis

As both NetApp#3 and NetApp #2 share the same analysis, a detailed analysis is given in the previous NetApp #2 analysis section, where a flow analysis of all feasible data flows between NetApp #2, NetApp#3, and possibly other connected instances such as cross-layer NetApps, ITS stations, and UEs were analyzed considering the four data sources:

- External NetApp or any Third-Party Cloud
- NetApp #3 (Cooperative C-ITS-S NetApp)
- NetApp #2 (Virtual RSU NetApp)
- Vehicle ITS Station (V-ITS-S)

#### 3.3.2.1 NSDs

Central C-ITS NSD	
Number of Services (NSDs)	1
Number of total VNFs	2
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information
How is it operated	Configuration is to be done manually for the moment
Dependencies	No from other NetApps

Table 3.3.1: Virtual Cooperative ITS-S NetApp NSD

### 3.3.2.2 VNFs

VNF Central C-ITS (NSD1)	
Number of total VNFs	2
Packaging Info:	VNF (OSM10)
Placement (latency from-to)	400-500ms
Internet access	Yes
Resource req/limits	1CPU, 1GB RAM, 5GB HDD
Delivery model	VM Image

Table 3.3.2: Virtual Cooperative ITS-S NetApp VNF

### 3.3.2.3 NSD and VNF relations/dependencies

In the same way, the NetApp #3 (C-ITS) consists of a single NS that encompasses 2 VNFs. It will gather information retrieved by the vRSU instance and make the link with other external cloud instances.

In general, NetApp #3 dependencies of the 5G System (requires slicing, core functionality, etc.):

- No direct connection to 5G radio resources
- May need slicing and specific QoS to perform low latency and time-critical communications based on 5G/4G/3G data
- Orchestrated dynamic VM deployment and MANO connectivity to recover deployment status information

### 3.3.3 NetApp definition

NetApp #3 (C-ITS-S) definition:

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Virtual Machine with containers running inside
  - Computing requirements depending on the number of served UE and the size of the zone to cover and (1-10+vCPU, 1-64+GB RAM, 10-100+GB Storage)
  - Three network interfaces supporting IP(v4/v6) communications
    - Virtual Machine with containers running inside
    - Computing requirements depending on the number of served UE and the size of the zone to cover and (1-10+vCPU, 1-64+GB RAM, 10-100+GB Storage)
    - Three network interfaces supporting IP(v4/v6) communications
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs:
  - SNMP metrics provided
  - Control and configuration through YoGoKo API
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - No direct connection to 5G radio resources
  - May need slicing and specific QoS to perform low latency and time-critical communications based on 5G/4G/3G data

- Orchestrated dynamic VM deployment and MANO connectivity to recover deployment status information
- How is it operated (and does it require manual interventions):
  - The Central C-ITS Station is deployed once in the 5G infrastructure and covers a wide geographic area which corresponds to the theoretical radio coverage of the whole 5G infrastructure.
  - A Central C-ITS station may be operated alone or along with vRSUs deployed on the edge to offload processing and operate local radio resources.
  - The Central C-ITS Station uses its downlink interface to communicate with vRSUs or UEs (e.g. vehicles, connected infrastructure or nomadic devices) in the coverage of the 5G infrastructure
  - The Central C-ITS Station uses its uplink interface to exchange information to third-party servers or other NetApps implementing C-ITS use cases
  - The Central C-ITS Station uses its management interface for control-plane operations such as orchestration or communication of metrics
  - The Central C-ITS Station is deployed unconfigured and configuration is managed through YoGoKo configuration API
  - The exchange of data between UE and the Central C-ITS Station must comply with C-ITS standards (data formats, communication protocols, security, etc.)
  - Once the infrastructure is in place, any type of service (able to exploit data transmitted between vehicles and the roadside) could be developed either to demonstrate the benefit of Central C-ITS Station for existing C-ITS services or to demonstrate how new C-ITS services can be developed
- Dependencies (does it expose or consume services from/to other NetApps): Yes, Efficient MEC handover NetApp, vOBU NetApp, vRSU NetApp and Multi-domain Migration NetApp.

C-ITS-S NETAPP'S NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	100.000 kbps
Uplink maximum throughput per UE	1.000 kbps
Isolation level	Virtual resources isolation
V2X Communication mode	No
Slice quality of service parameters: 3GPP 5QI	84 (for time-critical road safety services)
Maximum Packet Loss Rate	1%
Supported Device Velocity	250 km/h

Table 3.3.3: C-ITS NetApp (NetApp #3) NEST

### 3.3.3.1 NSDs definitions

Since both NetApp #2 and NetApp #3 share similar basic functionalities and rely on the ITS-S core code, YoGoKo uses the same PoC basic NSD and VNFs, where further detailed NSDs and VNFs are to be developed to identify unique use-case and functions specifics.

### 3.3.3.2 VNFs definitions

NetApp #2 and NetApp #3 have similar VNF descriptors as described in section 3.2.3 for the vRSU PoC, it is currently under development to adapt the OSM 10.

### 3.3.4 NetApp's tests plan and definition

The ongoing test definitions consider the associated functionalities tests and follow the guidelines and test plane detailed in WP5. The NetApp will be tested on the mandatory tests definitions 4 and 1. After identifying the test cases in D5.3 [3], a test descriptor will be developed accordingly.

The Central C-ITS Station is meant to be deployed on the core network and has no direct usage of radio resources. All communications come to the Central C-ITS Station under the form of IP(v4/v6) packets.

The Central C-ITS Station connects with NetApps in the 5G infrastructure and with third-party servers, tests may need to consider testing interconnection.

Since the Central C-ITS Station is responsible for a wide geographic zone, tests will need to bring special attention to scalability and proper operation under a high number of connected UEs, a high number of events, or a high network load. Tests will need to verify that the Central C-ITS station maintains its properties (e.g., latency, availability) under such situations.

Examples of test cases:

- A UE sends a message, this message must be transmitted to other UEs and to third-party servers or other NetApps deployed on the core.
- A third-party server sends a message, this message must be transmitted to other UEs and to other NetApps deployed on the core.
- With a number of connected UEs sending messages per second, a UE sends a message, this message must be transmitted to other UEs and to third-party servers or other NetApps deployed on the core under number of milliseconds.

The following table summarizes a set of service basic test examples for the C-ITS-S NetApp (NetApp #3):

ID	Definition	Test point	Expected value	Test method
1	A C-ITS message originated from the UE is received by other nodes	Application Plane	PASS/FAIL	Automated or Manual
2	A C-ITS message originated from an External server is received by other nodes	Application Plane	PASS/FAIL	Automated or Manual
3	Under high load, a C-ITS message originated from the UE is received by other nodes in less than a given time	Application Plane	PASS/FAIL	Automated or Manual

Table 3.3.4: Service test examples for the C-ITS NetApps (NetApp #3)

### 3.3.5 NetApp requirements for testbeds

Tests for the Central C-ITS Station do not rely on any radio requirement and will be fully virtualized. However, test cases may need to generate a high load and a high number of connected stations, which may require a big number of containers to simulate this load. HW requirements:

- A high number of containers (100-1000+)

## 3.4. NetApp #4

### 3.4.1 NetApp functional overview, current status and next steps

OdinS presents a solution that provides interdomain mobility capabilities to the vOBUs introduced in NetApp #1. This NetApp allows the vOBUs to be migrated to the MEC, closest to the real vehicle, reaching the low latencies requisites that characterize vehicular applications. This migration procedure ensures that the physical OBU will maintain connectivity with its virtual counterpart in the former domain while the new virtual instantiation is getting ready with the same configuration and state. Once it is ready, the data paths are updated to start using the new one without any packet loss.

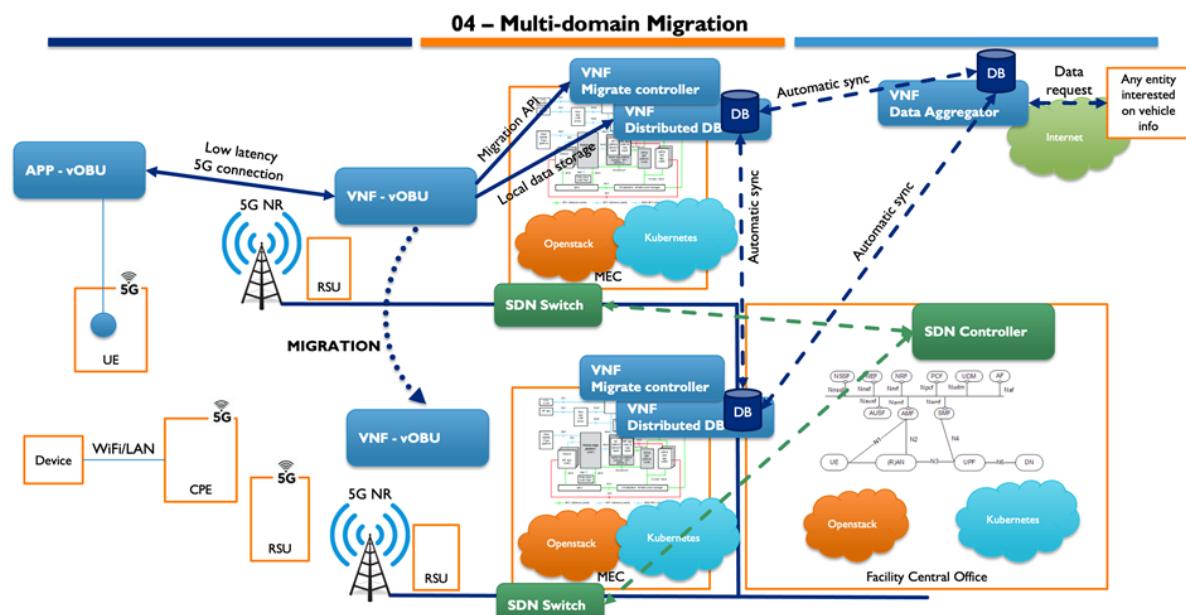


Figure 3.4.1: Multi-domain Migration NetApp architecture

The NetApp initial beta version is still in design with descriptors available for earlier OSM version (<8) and ongoing development for OSM10. The NetApp and the NEST are available and planned for further testing of functionality. The aim is to have the new version ready by end of Q3).

As aforementioned, some considerations about the NetApp functionality are still being discussed in OdinS. The main one is the possibility to unify this NetApp with the vOBU NetApp, as they share common entities, and they can provide the same service but being only one

entity. Therefore, in the next months, the integration of both NetApps will be studied and the implementation of the vOBU will be extended with the required functionality and the entities for the domain migration operation will be included. That is, the Distributed Database VNF will be adopted by NetApp #1, the Migrate Controller VNF functionalities will be integrated in the vOBU Manager VNF and the vOBU VNF is exactly the same implementation as in the vOBU NetApp. The final result of this integration will be shared with the consortium during the development of the project and reflected in the final design of the NetApps in D4.4 (M33).

### 3.4.2 NetApp analysis

In this subsection, the NSDs and VNFDs of the NetApp are presented from the point of view of the analysis. Note that these descriptors are the ones designed initially and will be changed in the integration of this NetApp with the vOBU NetApp.

#### 3.4.2.1 NSDs

Multi Domain Migration	
Number of Services (NSDs)	1
Number of total VNFs	3
Dependencies of the 5G System	Requires OpenFlow as part of the datapath for transparent migration and interaction with the SDN controllers of the infrastructure
How is it operated	Requires previous mapping between OpenFlow ports and VIM deployment. Protocol used: IPv6 (CoAP)
Dependencies	Yes: vOBU

Table 3.4.1: Multi Domain Migration NSD

#### 3.4.2.2 VNFs

vOBU (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM)
Placement (latency from-to)	30ms migration time
Internet access	No
Resource req/limits	2CPU, 1GB RAM, 5 GB HDD
Delivery model	VM Image

Table 3.4.2: vOBU VNF

Distributed DB (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM)

Placement (latency from-to)	30ms migration time
Internet access	No
Resource req/limits	2CPU, 1GB RAM, 40 GB HDD
Delivery model	VM Image

Table 3.4.3: Distributed DB VNF

Migrate Controller (NSD1)	
Number of total VNFs	3
Packaging Info:	VNF (OSM)
Placement (latency from-to)	30ms migration time
Internet access	No
Resource req/limits	2CPU, 1GB RAM, 5 GB HDD
Delivery model	VM Image

Table 3.4.4: Migrate Controller VNF

### 3.4.2.3 NSD and VNF relations/dependencies

The relationship among the different VNFs of the NetApp can be seen clearly in Figure 3.4.1. The NetApp consists in a single NS that encompasses the three VNFs. The vOBU interacts with the Controller to handle the instantiations and the migration and uses the Distributed DB as a local data storage. Therefore, the vOBU is dependent on the Controller and on the DB.

### 3.4.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - NSD, 3 VM Roles (VNFD): vOBU, Distributed DB and the vOBU.
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: No.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires dynamic VM deployment and MANO connectivity to recover deployment status information. It also requires interaction with the SDN controller.
- How is it operated (and does it require manual interventions):
  - Configuration of Mobile device requires human intervention. Right now, configuration with Ansible scripts, adaptation to fully automated deployment planned. Protocol used: IPv6 (CoAP).

Dependencies (does it expose or consume services from/to other NetApps): Yes, vOBU.

Multi-domain NETAPP'S NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	-

Uplink maximum throughput per UE	-
Isolation level	Virtual resources isolation
V2X Communication mode	Yes, New Radio (NR)
Slice quality of service parameters: 3GPP 5QI	9
Maximum Packet Loss Rate	1%
Supported Device Velocity	3: Vehicular: 10 km/h to 120 km/h

Table 3.4.5: Multi-domain NetApp's NEST

#### 3.4.3.1 NSDs definitions

NSD is available in an old version of OSM (<8), which is not supported by the project, it is currently being adapted to a newer version (OSM 10).

#### 3.4.3.2 VNFs definitions

VNFs are available in an old version of OSM (<8), which is not supported by the project, they are currently being adapted to a newer version (OSM 10).

#### 3.4.4 NetApp's tests plan and definition

The Multi-domain Migration NetApp test plan is very similar as the one presented for vOBU NetApp, as they share the same principles, and they have very similar entities. As mentioned in the initial section, the idea is to integrate this NetApp into the vOBU, so the future test plan will be almost equal, but adding some migration-specific functional testing.

The NetApp will follow the guidelines provided by WP5 [3] to elaborate the testplan. In WP5, the initial testplan is led by the NetApp definitions that were established in WP2. In this way, these tests are defined to check if a certain NetApp complies with the NetApp general definition. For this NetApp, the selected mandatory tests are the ones related to definitions 1, 3, 4 and 19; and the chosen optional tests are related to definitions 2, 6, 8, 13, 15 and 18.

Furthermore, the NetApp also executes performance tests to check that the infrastructure is working properly, and it can support the operation of the vOBUs. These tests check the end-to-end latency, packet delivery ratio, and jitter.

Regarding the functional testing of the NetApp, it is currently being designed, although it will reuse some of the principles stated in the vOBU tests.

The test descriptor will be developed to include all these tests by defining the different test cases and their parameters as well as the execution order and results collection. The tests will be implemented in Python scripts, which will feed the Robot framework. The custom test VNFs will be prepared based on Debian images.

To allow the testing of the Multi-domain Migration NetApp there are two main hardware/physical conditions. The first one is to evaluate the operation of the NetApp with a

static physical OBU, and the second one is with a moving physical OBU (onboarded in a car). Besides, it is necessary to have available two different domains to perform the migration process. Regarding the network conditions, the NetApp can be tested in isolation (no external traffic), or in a shared medium with heavy traffic that affects the network performance.

All the testing aspects discussed will be collected in the test plan and in the test descriptor. The final outcome of the testing phase will be given by the successful passing of all the aforementioned test cases.

### 3.4.5 NetApp requirements for testbeds

The Multi-domain Migration NetApp main requirement for the testbed is the same as the vOBU NetApp, the physical OBU, which must be present in the testbeds capable of hosting the NetApp. This physical OBU does not need high-end capabilities, it can be similar to a Raspberry Pi 4, and it must include a 5G modem to be able to connect to the network. Optionally, although desired, it should include an 802.11p [9] modem to experiment with multi-RAT access. The OBU client software is made in Python, and it will be provided to the testbeds in the form of a Docker container ready to deploy. Finally, the physical OBU should be connected to sensors of a vehicle to gather automotive info about the NetApp functioning. However, as an initial approach, this information can be substituted by dummy data produced by the client software.

Another important aspect to consider for the functioning of this NetApp is that it needs SDN flow paths to operate, in order to dynamically adapt the data flows and perform the migration without losing any packet. In consequence, the NetApp has to interact with the SDN controller of the domains to be able to react to the domain changes. Although this is the initial approach that was considered when developing the initial service, other options can be explored to do this, such as a dynamic allocation of a network slice. This is a work in progress issue that will be further detailed in future deliverables.

## 3.5. NetApp #5

### 3.5.1 NetApp functional overview, current status and next steps

Autonomous vehicles are usually equipped with multiple 4k cameras and sensors (e.g., Light Detection and Ranging (LIDAR) and Radio Detection and Ranging (RADAR)) that generate a huge amount of data to be transferred. Moreover, as autonomous vehicle technologies continue to evolve, the data generated inside cars only continues to grow exponentially. To enable the vehicle to be highly operational, there are remote/cloud services such as teleoperation, remote driving and vehicle remote assistance that use the generated data to both estimate the status of the vehicle and build an image of its surroundings. Such services usually send back the vehicle control commands and instructions to be executed instantaneously. Naturally, the success of these services depends on the communication latency and reliability.

The V2C R2C NetApp enables the vehicle to send and receive data in real-time (Figure 3.5.1). The NetApp is application-aware to ensure optimized data transmission based on the content being sent (e.g., real-time video or voice have higher priority than telemetry data). Since the transmission of the HD video generated by the cameras must be in real-time (i.e., buffering and retransmissions cannot be used) the NetApp uses techniques such as Forward-Error-Correction (FEC), channel bonding and dynamic encoding bitrate to ensure fast video delivery while maintaining maximum video quality. In addition, when multiple channels exist, the NetApp will prioritize the data delivered to the vehicle based on the performance of the channels. The following figures illustrate the V2C R2C NetApp architecture.

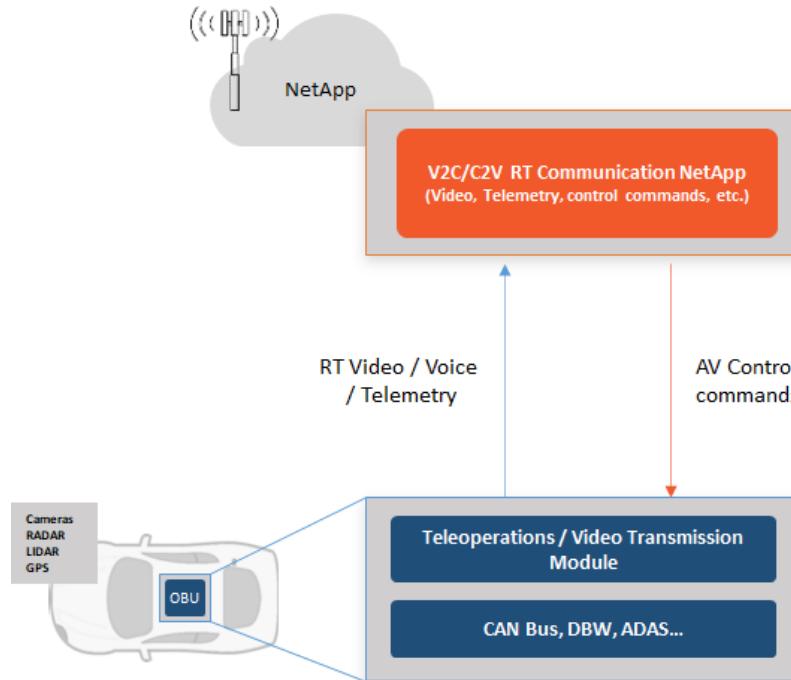


Figure 3.5.1: High-level architecture of Vehicle-to-Cloud Real-Time Communication (V2C R2C)

At the current stage, the NetApp is prepared and packaged as docker containers. The docker is built on the DriveU servers. The NetApp is deployed differently per various customer requirements and dependant on the customer equipment. DriveU will adjust the process for the specific testbed requirements.

#### Stage View

	Init & checkout	Determine version	Make	Build cameras	Test	Build debs	Upload build	Build images	Push new images
Average stage times: (Average full run time: ~31min 15s)	2min 16s	3s	3min 7s	18s	14min 36s	3min 52s	13s	4min 23s	39s
#347 Mar 16 17:19 1 commit	2min 22s	1s	3min 11s	18s	14min 42s	3min 32s	15s	9min 39s	9s
#346 Mar 16 10:39 1 commit	1min 57s	1s	3min 11s	18s	14min 40s	3min 41s	15s	30ms	27ms
#345 Mar 15 20:00 21 commits	1min 57s	3s	3min 25s	19s	14min 48s	3min 58s	14s	8min 43s	9s
#344 Mar 15 13:08 1 commit	2min 23s	4s	5min 11s	18s	14min 47s	5min 16s	19s	9min 37s	3min 19s
#343 Mar 14 22:09 No Changes	2min 5s	4s	5s failed						

Figure 3.5.2: Jenkins builder snapshot

Next step, deploy and configure at the specific testbed - the step will require some coordination with the testbed owners and is planned to be performed during the next 3 month (Q2-Q3 2022).

Test the deployment and adjust the configuration - the step is following the deployment step - will require additional coordination with the testbed owners and adaptation of the NetApp and the supporting services to the specific requirements and limitations. The planned time period is Q3-Q4 2022.

### 3.5.2 NetApp analysis

#### 3.5.2.1 NSDs

NetApp uses two Network Services, described with their respective Network Service Descriptors (NSDs). Network services' functionalities are described in the previous sections. Network Services:

- Cloud,
- Vehicle.

Cloud NSD	
Number of Services (NSDs)	1
Number of total VNFs	3-4 (depends on testbed configuration)
Dependencies of the 5G System	Either One slice URLLC or one slice WB (Wide-Band)
How is it operated	NetApp is working in the Vehicle client mode. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities and according to this it compresses and sends the data on the most efficient and reliable path.
Dependencies	None from other NetApps

Table 3.5.1: Cloud NSD

Vehicle NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Either One slice URLLC or one slice WB (Wide-Band)
How is it operated	NetApp is working in the Vehicle client mode. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities and according to this it compresses and sends the data on the most efficient and reliable path.
Dependencies	None from other NetApps

Table 3.5.2: Vehicle NSD

### 3.5.2.2 VNFs

VNF Vehicle side	
Packaging Info:	VNF/CNF
Placement (latency from-to)	Deployed using UE, Latency to server: max 40ms
Internet access	Optional
Resource req	2 vCPU, 4GB RAM, 20GB HDD, GPU (Nvidia)
Delivery model	Container

Table 3.5.3: Vehicle VNF

VNF Server	
Packaging Info:	VNF/CNF
Placement (latency from-to)	Latency from UE: max 40ms
Internet access	Yes
Resource req	2 vCPU, 4GB RAM, 20GB HDD, GPU (Nvidia)
Delivery model	Container

Table 3.5.4: Server VNF

### 3.5.2.3 NSD and VNF relations/dependencies

The NetApp is a single network service on a single slice (same slice for all VNFs in the app). The interconnection between the services is done over internal API which utilizes the UDP ports. Depending on a specific testbed configuration, it might be useful to work over TCP ports, therefore the connection between the various VNF will utilize several TCP and UDP ports. The specific ports will be coordinated during the integration or can be configurable as part of deployment automation engines.

### 3.5.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Virtual machine with custom image,
  - NSD/VNFD templates (OSM10).
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: Watchdog on a module since the application is mission critical.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - No clear dependency – the app is working over any communication technology with enough bandwidth and latency low enough to operate. 5G strives to bring the latency and the bandwidth to new levels, however it may be not at the same time and may introduce some limitation on connectivity.
- How is it operated (and does it require manual interventions): NetApp is working in the Vehicle client mode. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities and according to this it compresses and sends the data on the most efficient and reliable path.

- Dependencies (does it expose or consume services from/to other NetApps): No.

V2C R2C NETAPP's NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Isolation level	Virtual resources isolation
Downlink maximum throughput per UE	100000 kbps
Uplink maximum throughput per UE	1000 kbps
Slice quality of service parameters: 3GPP 5QI	9
Maximum Packet Loss Rate	10^-6
Supported device velocity	<120 km/h

Table 3.5.5: V2C R2C NetApp's NEST

### 3.5.3.1 NSDs definitions

```

nsd:
  nsd:
    - description: Vechicle_client_1
      designer: DriveU
      df:
        - id: default-df
          vnf-profile:
            - id: '1'
              virtual-link-connectivity:
                - constituent-cpd-id:
                  - constituent-base-element-id: '1'
                    constituent-cpd-id: mgmt-ext
                    virtual-link-profile-id: 5GASP
                  vnfd-id: Vechicle_client_1_knf
                id: Vechicle_client_1_ns
                name: Vechicle_client_1_ns
                version: '0.7'
              virtual-link-desc:
                - id: adminnet1
                  mgmt-network: 'true'
                vnfd-id:
                  - Vechicle_client_1_knf:

```

```

nsd:
  nsd:
    - description: Server_relay_1
      designer: DriveU
      df:
        - id: default-df
          vnf-profile:
            - id: '1'
              virtual-link-connectivity:
                - constituent-cpd-id:
                  - constituent-base-element-id: '1'
                    constituent-cpd-id: mgmt-ext
                    virtual-link-profile-id: 5GASP
                  vnfd-id: Server_relay_1_knf
                id: Server_relay_1_ns
                name: Server_relay_1_ns

```

```

version: '0.7'
virtual-link-desc:
- id: adminnet1
  mgmt-network: 'true'
vnfd-id:
- Server_relay_1_knf:

```

### 3.5.3.2 VNFs definitions

```

vnfd:
description:Vehicle side Streamer and commands receiver
df:
- id:default-df
ext-cpd:
- id:mgmt-ext
  k8s-cluster-net:mgmtnet
id: Vehicle
k8s-cluster:
nets:
- id:mgmtnet
kdu:
- name: NetApp_5_Vehicle
  helm-chart:
  mgmt-cp:mgmt-ext
product-name:NetApp5_vehicle
provider:DriveU
version:'0.7'

```

```

vnfd:
description:Server Relay side and command Stream
df:
- id:default-df
ext-cpd:
- id:mgmt-ext
  k8s-cluster-net:mgmtnet
id: Server_Relay
k8s-cluster:
nets:
- id:mgmtnet
kdu:
- name: NetApp_5_Server
  helm-chart:
  mgmt-cp:mgmt-ext
product-name:NetApp5_Server
provider:DriveU
version:'0.7'

```

### 3.5.4 NetApp's tests plan and definition

The NetApp will follow the test plan defined in WP5 [3]:

- mandatory test cases: 2, 9, 15, 19
- optional test cases: 6, 8, 11, 14

The detailed test plan specific for the NetApp:

- Five modules are deployed and running
  - Streamer – generates the video stream and encodes it for tests
  - Connectador – provides streamer connectivity layer to 5G modem
  - Relay – receives the network packets and controls the network behaviors
  - Node – controls the video stream

- Reporting service – logging database service (normally AWS S3)
- Test list:
  - The basis is the system initiation – the modules are up and reporting as running
  - The streamer and connectador are connected to relay and node modules and reporting as connected
  - The system is synchronized using the cellular network
  - Connectivity to reporting service is established
  - Beginning of the video stream – the streamer is generating a video pattern pushed through the streamer and 5G system to the relay and node modules (terminated and measured). The video BW is configured to 10Mbps (encoded) and the modem BW is configured to 10Mbps. The metric list includes:
    - Metric list
      - Disk usage
        - Streamer
        - Relay
      - Latency
        - Video latency – frame by frame reported in an external report – graphical and tabular format
        - Packet latency reported as an external report – graphical and tabular format
      - BW
        - Reported on a frame time aggregation basis
      - Loss
        - Reported on a frame time aggregation basis
  - Extensive tests
    - Video BW is being changed according to the predefined pattern (between 2Mbps to 20Mbps)
    - Latency
      - Video latency – frame by frame reported in an external report – graphical and tabular format
      - Packet latency reported as an external report – graphical and tabular format
    - BW
      - Reported on a frame time aggregation basis
    - Modem BW is being changed according to the predefined pattern (between 2Mbps to 20Mbps)
    - Latency
      - Video latency – frame by frame reported in an external report – graphical and tabular format
      - Packet latency reported as an external report – graphical and tabular format
      - BW
        - Reported on a frame time aggregation basis
      - Loss
        - Reported on a frame time aggregation basis

- The KPI list is changing as we progress with the 5G deployment – the current status is:
  - Overall, at any test, the frame loss should be under 50 frame skips per hour (this can be normalized and doesn't need to run for 1 hour)
  - The skips should be correlated with the network loss, latency, and BW – meaning that the system is built to work with the unreliable network but up to specific conditions. In case of loss greater than the system specifications, the frame will be lost
    - Meaning that the performance KPI here is not a hard value but depends on the network
    - The network performance is under the service provider's responsibility
    - The suggested performance for LTE network
      - Loss < 1%
      - BW > 1Mbps (uplink only)
      - Latency < 90ms (uplink only)
    - For 5G networks, the suggestions are better

### 3.5.5 NetApp requirements for testbeds

The main requirements are:

- Client side:
  - GPU enabled computer (can be Jetson or PC)
  - 5G modem
  - Ubuntu 18.04 or higher (not custom)
- Server side:
  - GPU enabled (Nvidia)
  - Connection to external reporting service (can be internal at later stages)

## 3.6. NetApp #6

### 3.6.1 NetApp functional overview, current status and next steps

There is an industry consensus today that autonomous vehicles will need help in making decisions, especially in unusual, dangerous situations that can happen on the road and may require violating the traffic laws (e.g., crossing double yellow lines). For these cases, the industry has started to adopt teleoperation/remote driving solutions that enable a remote human operator to monitor and take control over the car, if needed.

DriveU and BLB present here a solution that enables a remote operator to take full/partial control over an autonomous vehicle in unusual/dangerous situations that can happen on the road (e.g., let an autonomous vehicle crossing double yellow lines). Ensuring safe teleoperation and human remote assistance entails reliable transmission of high-quality real-time video with minimum latency.

In this use case, we will equip a virtual vehicle instance with one or more 5G modems and a software (SW) module that enables the transmission of HD video to the tele-operation centre

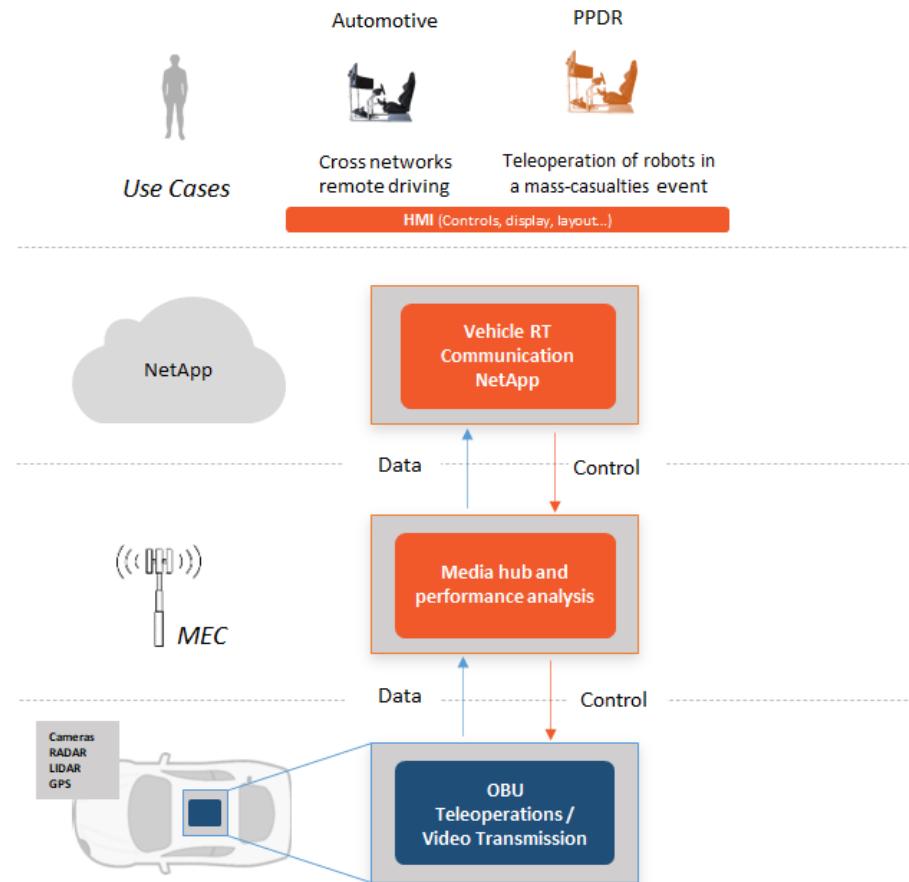


Figure 3.6.1: Remote Human Driving NetApp - Teleoperation for assisting vehicles in complex situations

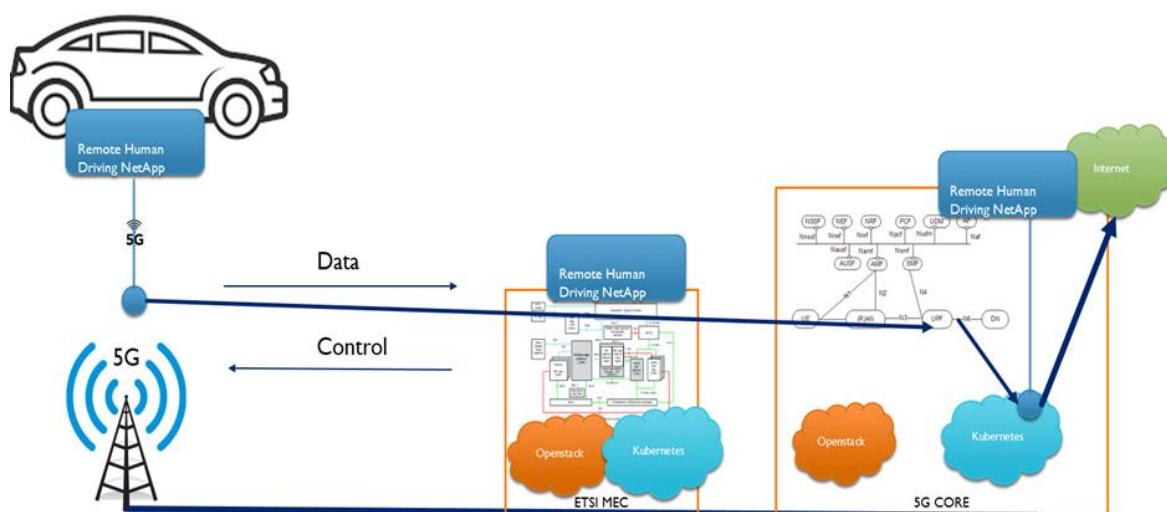


Figure 3.6.2: Vehicle-to-Cloud Real-Time Communication (V2C R2C) NetApp.

At the current stage, the NetApp is prepared and packaged as docker containers. The docker is built on the DriveU servers. The NetApp is deployed differently per various customer requirements and dependent on the customer equipment. DriveU will adjust the process for the specific testbed requirements.

### 3.6.2 NetApp analysis

NetApp uses same functionalities as NetApp #5 with addition to the node service. Network Services:

- Cloud,
- Vehicle,
- Node.

As it is operated at the customer premisses, the Node can be connected using the 5G network or locally through LAN. At the current stage, it is suggested to use it locally as part of the Cloud service, therefore no additional NSD is required.

#### 3.6.2.1 NSDs

NetApp uses of three Network Services, described with their respective Network Service Descriptors (NSDs). Network services' functionalities are described in the previous sections. Network Services:

- Cloud,
- Vehicle,
- Node.

Cloud NSD	
Number of Services (NSDs)	1
Number of total VNFs	3-4 (depends on testbed configuration)
Dependencies of the 5G System	Either One slice URLLC ore one slice WB (WideBand)
How is it operated	Netapp is working in mode of the Vehicle client. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities, according to it compresses and send the data at the most efficient and reliable path.
Dependencies	None from other netapps

Table 3.6.1: Cloud NSD

Vehicle NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Either One slice URLLC ore one slice WB (WideBand)
How is it operated	Netapp is working in mode of the Vehicle client. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities, according to it compresses and send the data at the most efficient and reliable path.
Dependencies	None from other netapps

Table 3.6.2: Vehicle NSD

Node NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Either One slice URLLC or one slice WB (WideBand)
How is it operated	Netapp is working in mode of the Vehicle server client with the relay and used as an interface to the server and to the drivers side
Dependencies	None from other netapps

Table 3.6.3: Cloud NSD

### 3.6.2.2 VNFs

VNF Vehicle side	
Packaging Info:	VNF/CNF
Placement (latency from-to)	Deployed using UE, Latency to server: max 40ms
Internet access	Optional
Resource req	2 vCPU, 4GB RAM, 20GB HDD, GPU (Nvidia)
Delivery model	Container

Table 3.6.4: VNF vehicle side

VNF Server	
Packaging Info:	VNF/CNF
Placement (latency from-to)	Latency from UE: max 40ms
Internet access	Yes
Resource req	2 vCPU, 4GB RAM, 20GB HDD, GPU (Nvidia)
Delivery model	Container

Table 3.6.5: VNF server

VNF Node	
Packaging Info:	VNF/CNF
Placement (latency from-to)	Latency from server: max 10ms
Internet access	Yes
Resource req	4 vCPU, 16GB RAM, 50GB HDD, GPU (Nvidia)
Delivery model	Container

Table 3.6.6: VNF Node

### 3.6.2.3 NSD and VNF relations/dependencies

The NetApp is a single network service on a single slice (same slice for all VNFs in the app). The interconnection between the services is done over internal API which utilizes the UDP ports. Depending on a specific testbed configuration, it might be useful to work over TCP ports, therefore the connection between the various VNF will utilize several TCP and UDP ports. The specific ports will be coordinated during the integration or can be configurable as part of deployment automation engines. The user (node) VNF is connected using external API which will be exposed to the local network.

### 3.6.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Virtual machine with custom image,
  - NSD/VNFD templates (OSM10).
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: Watchdog on a module since the application is mission critical.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - No clear dependency – the app is working over any communication technology with enough BW and latency low enough to operate. 5G strives to bring the latency and the BW to a new level, however it may be not in the same time and may introduce some limitation on connectivity. Meaning it is a TBD.
- How is it operated (and does it require manual interventions): NetApp is working in mode of the Vehicle client. This mode provides an efficient use of the communication interfaces to the server by analyzing the interfaces capabilities, according to it compresses and send the data at the most efficient and reliable path.
- Dependencies (does it expose or consume services from/to other NetApps): No.

V2C R2C NETAPP's NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region specification	Murcia, Aveiro, Bristol, Bucharest
Isolation level	Virtual resources isolation
Downlink maximum throughput per UE	100000 kbps
Uplink maximum throughput per UE	1000 kbps
Slice quality of service parameters: 3GPP 5QI	9
Maximum Packet Loss Rate	10^-6
Supported device velocity	<120 km/h

Table 3.6.7: V2C R2C NetApp's NEST

### 3.6.3.1 NSDs definitions

```
nsd:  
nsd:  
- description: Vechicle_client_1  
designer: DriveU  
df:  
- id: default-df  
vnf-profile:  
- id: '1'  
virtual-link-connectivity:  
- constituent-cpd-id:  
- constituent-base-element-id: '1'  
constituent-cpd-id: mgmt-ext  
virtual-link-profile-id: 5GASP  
vnfd-id: Vechicle_client_1_knf  
id: Vechicle_client_1_ns  
name: Vechicle_client_1_ns  
version: '0.7'  
virtual-link-desc:  
- id: adminnet1  
mgmt-network: 'true'  
vnfd-id:  
- Vechicle_client_1_knf:
```

```
nsd:  
nsd:  
- description: Server_relay_1  
designer: DriveU  
df:  
- id: default-df  
vnf-profile:  
- id: '1'  
virtual-link-connectivity:  
- constituent-cpd-id:  
- constituent-base-element-id: '1'  
constituent-cpd-id: mgmt-ext  
virtual-link-profile-id: 5GASP  
vnfd-id: Server_relay_1_knf  
id: Server_relay_1_ns  
name: Server_relay_1_ns  
version: '0.7'  
virtual-link-desc:  
- id: adminnet1  
mgmt-network: 'true'  
vnfd-id:  
- Server_relay_1_knf:
```

```
nsd:  
nsd:  
- description: Node_1  
designer: DriveU  
df:  
- id: default-df  
vnf-profile:  
- id: '1'  
virtual-link-connectivity:  
- constituent-cpd-id:  
- constituent-base-element-id: '1'  
constituent-cpd-id: mgmt-ext  
virtual-link-profile-id: 5GASP  
vnfd-id: Node_1_knf  
id: Node_1_ns  
name: Node_1_ns  
version: '0.7'
```

```

virtual-link-desc:
- id: adminnet1
  mgmt-network: 'true'
vnfd-id:
- Node_1_knf:

```

### 3.6.3.2 VNFs definitions

```

vnfd:
description:Vehicle side Streamer and commands receiver
df:
- id:default-df
ext-cpd:
- id:mgmt-ext
  k8s-cluster-net:mgmtnet
id: Vehicle
k8s-cluster:
nets:
- id:mgmtnet
kdu:
- name: NetApp_6_Vehicle
  helm-chart:
  mgmt-cp:mgmt-ext
product-name:NetApp5_vehicle
provider:DriveU
version:'0.7'

```

```

vnfd:
description:Server Relay side and command Stream
df:
- id:default-df
ext-cpd:
- id:mgmt-ext
  k8s-cluster-net:mgmtnet
id: Server_Relay
k8s-cluster:
nets:
- id:mgmtnet
kdu:
- name: NetApp_6_Server
  helm-chart:
  mgmt-cp:mgmt-ext
product-name:NetApp5_Server
provider:DriveU
version:'0.7'

```

```

vnfd:
description:Display Node
df:
- id:default-df
ext-cpd:
- id:mgmt-ext
  k8s-cluster-net:mgmtnet
id: Display_Node
k8s-cluster:
nets:
- id:mgmtnet
kdu:
- name: NetApp_6_Node
  helm-chart:
  mgmt-cp:mgmt-ext
product-name:NetApp5_Server
provider:DriveU
version:'0.7'

```

### 3.6.4 NetApp's tests plan and definition

The NetApp will follow the test plan defined in WP5 [3]:

- mandatory test cases: 2, 9, 15, 19
- optional test cases: 6, 8, 11, 14

The detailed test plan specific for the NetApp:

- Five modules are deployed and running
  - Streamer – generating the video stream and encodes it for tests
  - Connectador – streamer connectivity layer to 5G modem
  - Relay – receives the network packets and controls the network behaviors
  - Node – control the video stream
  - Reporting service – logging database service (normally AWS S3)
- Test list:
  - The basic is the system initiation – the modules are up and reporting as running
  - The streamer and connectador are connected to relay and node and reporting as connected
  - The system is synchronized using the cellular network
  - Connectivity to reporting service is established
  - Beginning of the video stream – the streamer is generating a video pattern pushed through the streamer and 5G system to the relay and node (terminated and measured). The video BW configured to 10Mbps (encoded), modem BW configured to 10Mbps:
    - Metric list
      - Disk usage
        - Streamer
        - Relay
      - Latency
        - Video latency – frame by frame reported in an external report – graphical and tabular
        - Packet latency reported as an external report – graphical and tabular format
      - BW
        - Reported on a frame time aggregation basis
      - Loss
        - Reported on a frame time aggregation basis
    - Extensive tests
      - Video BW is being changed according to the predefined pattern (between 2Mbps to 20Mbps)
        - Latency
          - Video latency – frame by frame reported in an external report – graphical and tabular
          - Packet latency reported as an external report – graphical and tabular format
        - BW
          - Reported on a frame time aggregation basis

- Modem BW is being changed according to the predefined pattern (between 2Mbps to 20Mbps)
    - Latency
      - Video latency – frame by frame reported in an external report – graphical and tabular
      - Packet latency reported as an external report – graphical and tabular format
    - BW
      - Reported on a frame time aggregation basis
    - Loss
      - Reported on a frame time aggregation basis
- The KPI list is changing as we progress with the 5G deployment – the current status is:
  - Overall, at any test, the frame loss should be under 50 frame skips per hour (normalized – doesn't need to run for 1 hour)
  - The skips should be correlated with the network loss, latency, and BW – meaning that the system is built to work with the unreliable network but up to specific conditions. In case of loss greater than the system spec, the frame will be lost
    - Meaning that the performance KPI here is not a hard value but depends on the network
    - The network performance is under the service provider's responsibility
      - The suggested performance for LTE network
        - Loss < 1%
        - BW > 1Mbps (uplink only)
        - Latency < 90ms (uplink only)
      - For 5G networks, the suggestions are better

### 3.6.5 NetApp requirements for testbeds

The main requirements are:

- Client side:
  - GPU enabled computer (can be Jetson or PC)
  - 5G modem
  - Ubuntu 18.04 or higher (not custom)
- Server side:
  - GPU enabled (Nvidia)
  - Connection to external reporting service (can be internal at later stages)

### 3.7. NetApp #7

#### 3.7.1 NetApp functional overview, current status and next steps

UNIVBRIS is developing an Efficient MEC Handover (EMHO) NetApp, which is designed to support the other compatible NetApp(s) to improve their performance in the MEC environment. The EMHO NetApp is a Machine Learning based application which consumes the radio monitoring data and predicts the probability of UE being handed over from one access point to another in a radio access environment in the next n seconds.

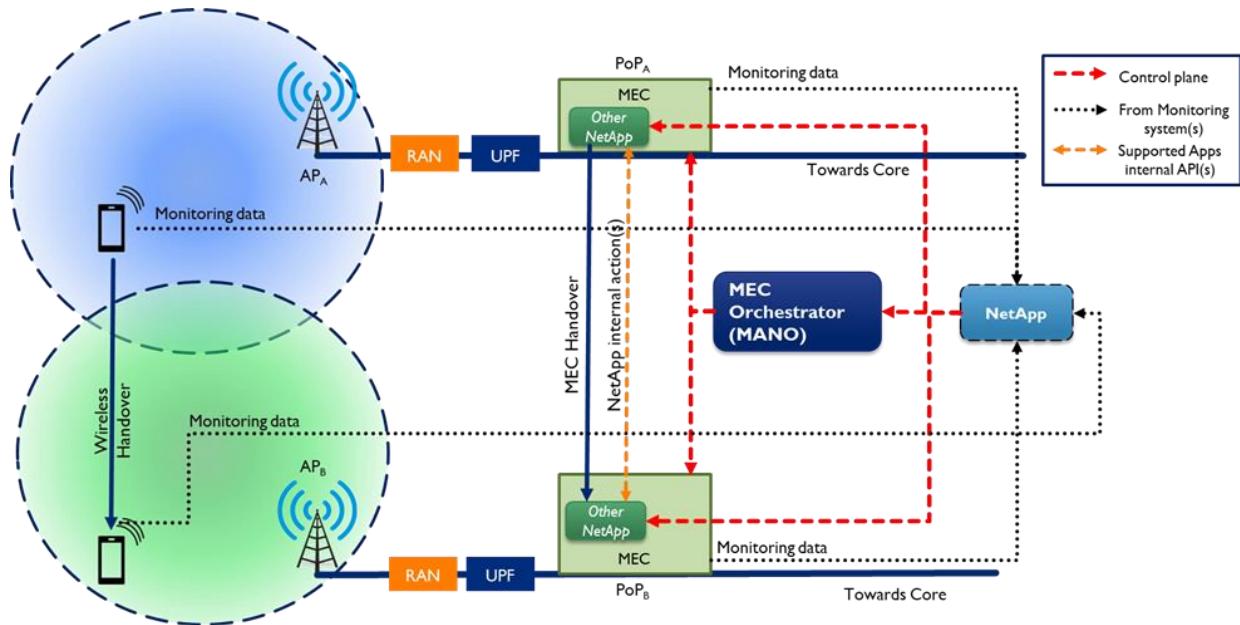


Figure 3.7.1: Efficient MEC Handover NetApp

The NetApp is in its initial version and has been deployed using OSM10 on Bristol's testbed. The NetApp's functionality has been tested and evaluated locally in a partially integrated environment. Currently, the NetApp has been deployed as VMs on the Openstack at the Bristol's testbed.

Furthermore, the NetApp is being amended to create the appropriate APIs to communicate with the supported NetApps. Also, the packaging of the NetApp is being changed to enable the deployment as KNFs on the Kubernetes environment.

The NetApp is onboarded on the Bristol's testbed using OSM as shown in the screenshot in Figure 3.7.2. The NetApp is onboarded as a VNF which can be verified using the screenshot in Figure 3.7.3 and relevant networks created in Openstack in Figure 3.7.4.



Dashboard > Projects > 5GASP\_UNIVBRIS > NS Instances

### NS Instances

Init running / configured failed scaling

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
Netapp7	078699f0-435a-412a-92af-e487d50eff23	5gasp_emho_ns	<span style="color: green;">✓</span>	<span style="color: green;">✓</span>	Done	<span style="color: blue;">Edit</span> <span style="color: red;">Delete</span> <span style="color: orange;">Actions</span>

Dashboard > Projects > 5GASP\_UNIVBRIS > VNF Instances

### VNF Instances

Identifier	VNFD	Member Index	NS	Created At	Actions
4361f3b2-a087-4734-afd7-830bf5d241a7	5gasp_emho_vnfd	emho	078699f0-435a-412a-92af-e487d50eff23	Mar-14-2022 12:16:33	<span style="color: blue;">VNFR</span> <span style="color: orange;">i</span>

Figure 3.7.2: EMHO On-boarded in OSM

## Instances

Displaying 13 items

INSTANCE ID	FILTER	LAUNCH INSTANCE	DELETE INSTANCES	MORE ACTIONS						
<input type="checkbox"/> Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
<input type="checkbox"/> Netapp7-emho-5gasp_emho_vnfd-VM-0	5gasp_emho	SGASP-Management 10.68.107.141 SGASP-Netapp 20.1.1.220 SGASP-Monit 15.1.1.61	5gasp_emho_vnfd-VM-flv-1	-	Active	wtc-az	None	Running	0 minutes	<span style="color: blue;">CREATE SNAPSHOT</span>

Figure 3.7.3: Instance of VNF in Openstack

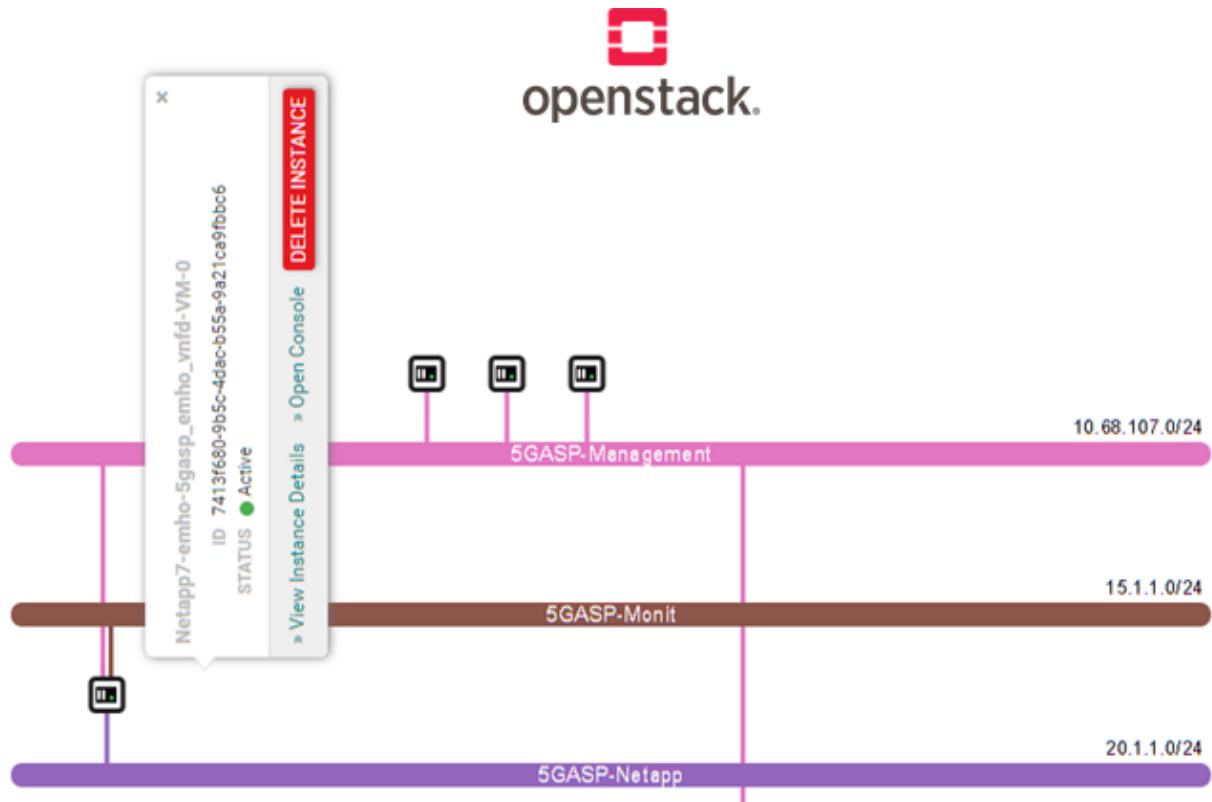


Figure 3.7.4: Relevant Networks in Openstack

### 3.7.2 NetApp analysis

The following subsection will cover the analysis of NSDs and VNFDs of the EMHO NetApp.

#### 3.7.2.1 NSDs

EMHO NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Monitoring information from the connected User Equipment (mobile phones)
How is it operated	REST APIs consumes the monitoring data from the UEs, and the NetApp predicts handover probability to all cells. The prediction is then provided to the supported NetApp(s) over REST APIs.
Dependencies	No

Table 3.7.1: EMHO NSD

#### 3.7.2.2 VNFs

VNF EMHO ML(NSD1)	
Number of total VNFs	1
Packaging Info:	VNF (OSM10)
Placement (latency from-to)	-
Internet access	No
Resource req/limits	8CPU, 10GB RAM, 40GB HDD
Delivery model	VM Image

Table 3.7.2: VNF EMHO ML

#### 3.7.2.3 NSD and VNF relations/dependencies

The EMHO NetApp will expose some APIs to get the monitoring data from the testbed and will communicate the predictions with the supported NetApp over a predefined interface. This is shown in Figure 3.7.5 which also includes the mapping of VNFD and NSD over the specific networks it will use for communication. The EMHO NetApp will be hosted in a MEC node alongside with the supported NetApp which can be hosted and/or moved across the nodes in the MEC environment.

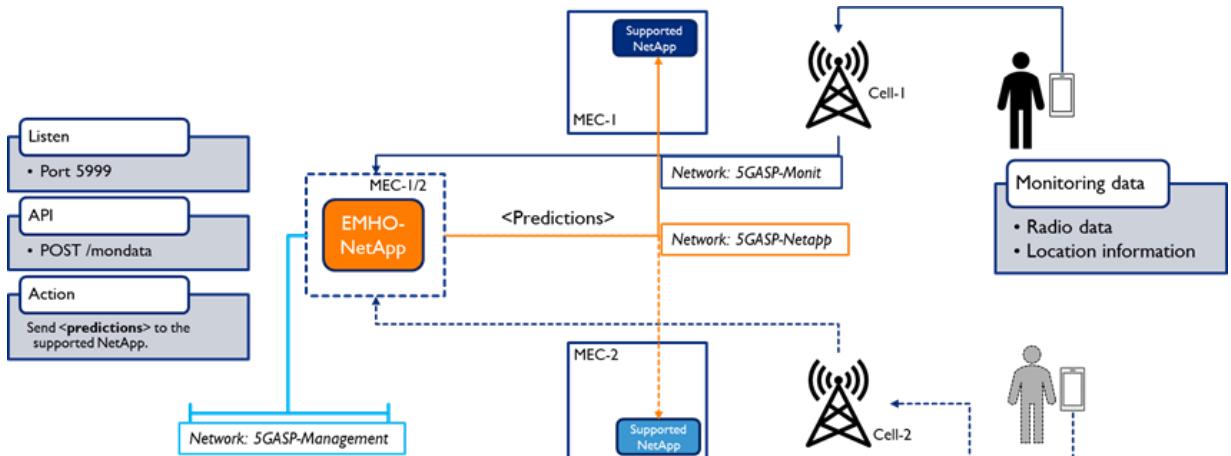


Figure 3.7.5: EMHO - NSD and VNFD relations

### 3.7.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - 1x NSD, 1x VNFD with trained Neural Network model hosting to predict the handover(s)
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: No.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires monitoring data from the UE/Network.
- How is it operated (and does it require manual interventions):
  - REST APIs consumes the monitoring data from the UEs, and the NetApp predicts the handover probability to all cells. The prediction is then provided to the supported NetApp(s) over REST APIs.
- Dependencies (does it expose or consume services from/to other NetApps):
  - It can support any other NetApp with appropriate API to consume the handover predictions.

EMHO NETAPP'S NEST	
Area of service	UK, GR
Area of service: Region specification	Bristol, Patras
Downlink maximum throughput per UE	-
Uplink maximum throughput per UE	-
Isolation level	Virtual resources isolation
V2X Communication mode	-
Slice quality of service parameters: 3GPP 5QI	5 (Can change as per the supported NetApp requirements)
Maximum Packet Loss Rate	1%
Supported Device Velocity	0-5 kmph

Table 3.7.3: EMHO NetApp's NEST

### 3.7.3.1 NSDs definitions

```
nsd:  
  nsd:  
    - description: Netapp#7 Efficient Handover Netapp NS  
    df:  
      - id: 5gasp_emho_ns  
      vnf-profile:  
        - id: 'emho'  
        virtual-link-connectivity:  
          - constituent-cpd-id:  
            - constituent-base-element-id: '1'  
              constituent-cpd-id: vnf-cp0-ext  
              virtual-link-profile-id: mgmtnet  
          - constituent-cpd-id:  
            - constituent-base-element-id: '1'  
              constituent-cpd-id: vnf-cp1-ext  
              virtual-link-profile-id: netappnet  
          - constituent-cpd-id:  
            - constituent-base-element-id: '1'  
              constituent-cpd-id: vnf-cp2-ext  
              virtual-link-profile-id: monitnet  
              vnfd-id: 5gasp_emho_vnfd  
        name: 5gasp_emho_ns  
        id: '5gasp_emho_ns'  
        version: '1.0'  
        virtual-link-desc:  
          - id: mgmtnet  
            mgmt-network: true  
            vim-network-name: 5GASP-Management  
          - id: monitnet  
            mgmt-network: true  
            vim-network-name: 5GASP-Monit  
          - id: netappnet  
            mgmt-network: true  
            vim-network-name: 5GASP-Netapp  
            vnfd-id:  
              - 5gasp_emho_vnfd
```

### 3.7.3.2 VNFs definitions

```
vnfd:  
  description: Generated by OSM package generator  
  df:  
    - id: default-df  
    instantiation-level:  
      - id: default-instantiation-level  
      vdu-level:  
        - number-of-instances: 1  
          vdu-id: 5gasp_emho_vnfd-VM  
    vdu-profile:  
      - id: 5gasp_emho_vnfd-VM  
        min-number-of-instances: 1  
  ext-cpd:  
    - id: vnf-cp0-ext  
    int-cpd:  
      cpd: eth0-int  
      vdu-id: 5gasp_emho_vnfd-VM  
    - id: vnf-cp1-ext  
    int-cpd:  
      cpd: eth1-int  
      vdu-id: 5gasp_emho_vnfd-VM  
    - id: vnf-cp2-ext
```

```

int-cpd:
  cpd: eth2-int
  vdu-id: 5gasp_emho_vnfd-VM
id: 5gasp_emho_vnfd
mgmt-cp: vnf-cp0-ext
product-name: 5gasp_emho_vnfd
provider: 'University of Bristol'
sw-image-desc:
  - id: 5gasp_emho
    image: 5gasp_emho
    name: 5gasp_emho
vdu:
  - description: 5gasp_emho_vnfd-VM
    id: 5gasp_emho_vnfd-VM
    int-cpd:
      - id: eth0-int
        virtual-network-interface-requirement:
          - name: eth0
            position: 1
            virtual-interface:
              type: PARAVIRT
      - id: eth1-int
        virtual-network-interface-requirement:
          - name: eth1
            position: 2
            virtual-interface:
              type: PARAVIRT
      - id: eth2-int
        virtual-network-interface-requirement:
          - name: eth2
            position: 3
            virtual-interface:
              type: PARAVIRT
      name: 5gasp_emho_vnfd-VM
      sw-image-desc: 5gasp_emho
      virtual-compute-desc: 5gasp_emho_vnfd-VM-compute
      virtual-storage-desc:
        - 5gasp_emho_vnfd-VM-storage
    version: '1.0'
    virtual-compute-desc:
      - id: 5gasp_emho_vnfd-VM-compute
        virtual-cpu:
          num-virtual-cpu: 8
        virtual-memory:
          size: '10'
    virtual-storage-desc:
      - id: 5gasp_emho_vnfd-VM-storage
        size-of-storage: '40'

```

### 3.7.4 NetApp's tests plan and definition

The EMHO NetApp will follow the test plan guidelines as described in the WP5 [3]. It will verify if the NetApp follows the guidelines. The NetApp will be tested on the mandatory test definitions 4 and 19. Additionally, on the optional tests listed in definition 2.

The EMHO NetApp will also have a few functional and performance related tests:

**Performance Tests:** These tests will be carried out using a special VM/container which will act as an agent to gather the performance parameters as per the requirements of the supported NetApp(s). The important KPIs for the NetApp are: availability, packet loss and the response time.

- The NetApp is required to be available for more than 95% of time as the downtime might affect the supported NetApp. However, the availability requirement can be dictated by the supported NetApp as per the latter's SLA. The testing component(s) will monitor the availability using the bash or python scripts remotely.
- The packet loss during the transmission of monitoring data can severely impact the handover predictions and the prediction information loss can impact the operation of the supported NetApp.
- Another test will monitor the response time between data availability and the prediction output to the supported NetApp. The actual minimum response time will be dictated by the supported NetApp to meet its SLA.

**Functional tests:** The following tests would verify the functionality of the EMHO NetApp:

- The prediction should have the probability of handover to each of the cells available in the testbed.
- The APIs and ports should be available for receiving the monitoring data and to send out the predictions.

**Procedure:** The test container will be hosted in the same network as the UE to perform the aforementioned tests as shown in the Figure 3.7.6.

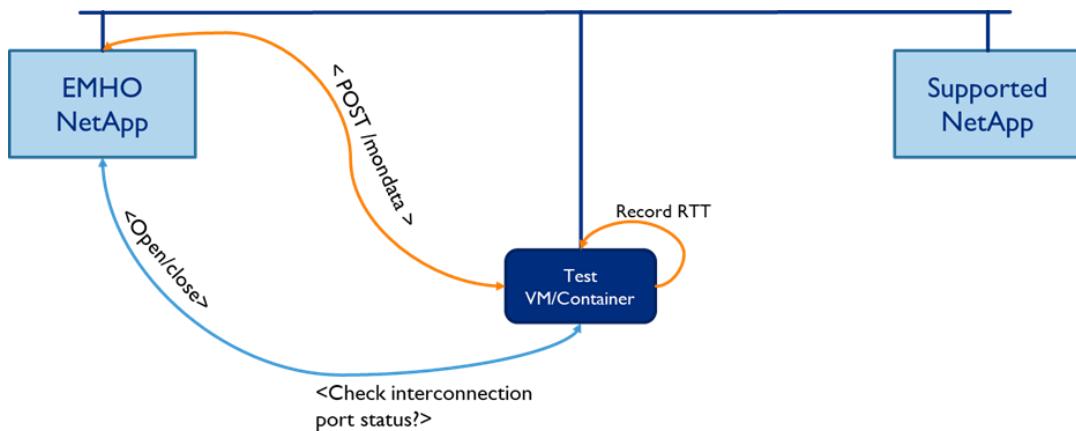


Figure 3.7.6: EMHO - testing procedure

### 3.7.5 NetApp requirements for testbeds

The EMHO NetApp would need a virtual environment (VM or container) to run and would depend on the monitoring data from the UE operating in a 5G network. Following are the requirements of the NetApp from the testbed.

- Hardware:
  - User Equipment (mobile phones)
    - Access to 5G network in the testbed.
    - Preferred OS: Android
- EMHO NetApp deployment:
  - Virtual environment: VM or containers.
  - Edge compute nodes connected to the Radio Access points. (1:1 mapping).
- Monitoring data:
  - Radio monitoring data from the UE.

## 3.8. NetApp #8

### 3.8.1 NetApp functional overview, current status and next steps

PrivacyAnalyzer offers the functionality for aiding the 5G/IoT integrators to assess the privacy strength of their applications or services against confidential information disclosure before they can release their solutions to their clients. PrivacyAnalyzer receives testing sessions' traffic between the devices (UEs, IoT devices, etc.) participating in the integrator's test sessions, scans the sessions' message contents for privacy sensitive content and reports the outcomes of the analysis to the end-user (integrator) with the goal to enable the end-user to resolve the privacy issues detected in the test sessions before releasing her application or service to her client, typically a network/service operator or an IoT service provider.

Within the context of 5GASP, PrivacyAnalyzer shall be used to assess the privacy strength of the network messages stemming from 5G IOPS UEs (see NetApp #9) carried by First Responders and this applies to the PPDR use case of WP4.

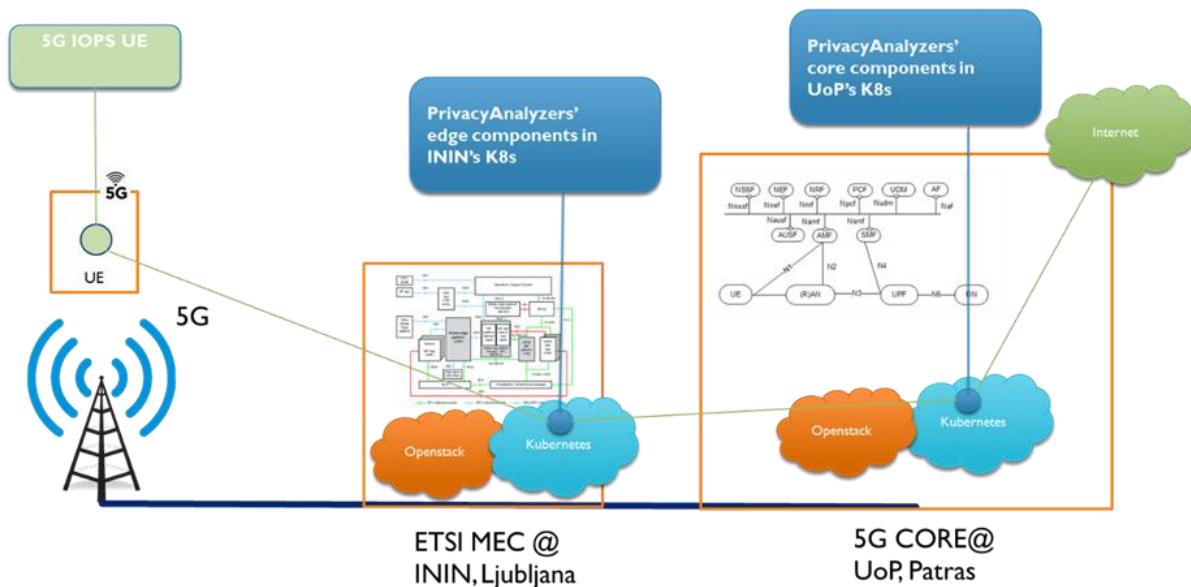


Figure 3.8.1: PrivacyAnalyzer's deployment in the ININ and UoP testbeds with the NetApp analyzing the privacy strength of networkd messages from 5G IOPS UEs in the PPDR use

Figure 3.8.2 displays the successful instantiation of PrivacyAnalyzer's demonstration Network Service within our local OSM 10 instance.

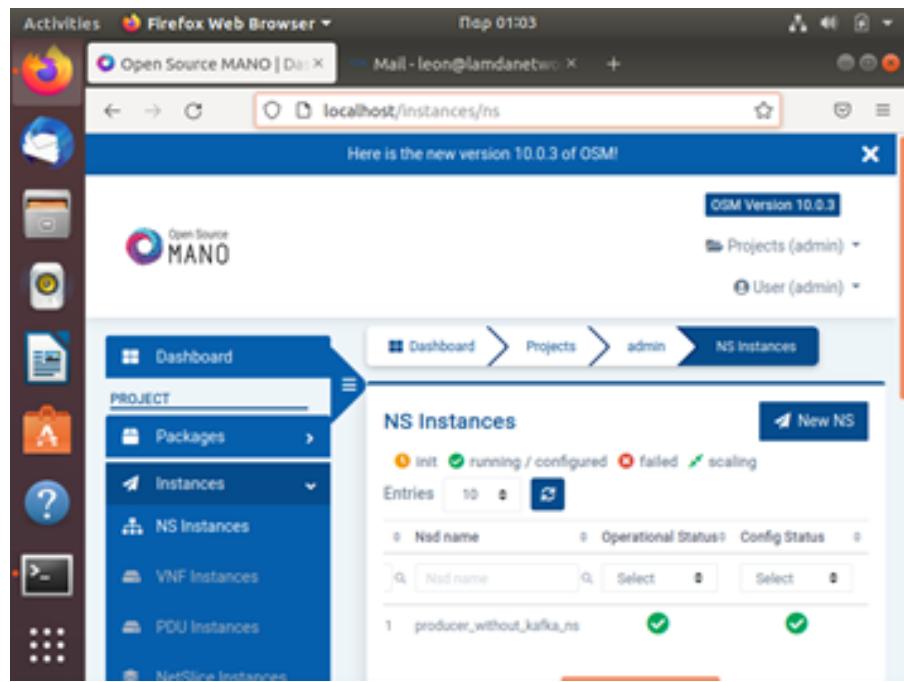


Figure 3.8.2: Local-OSM onboarding of the PrivacyAnalyzer demo

The following will provide some screenshots from a local demonstration which we also gave as a live demo to the 5GASP partners during a regular WP4 weekly call in March 2022. In this demonstration, our software is deployed on OSM as depicted in the previous figure and we run our docker images in our own K8s deployment. PrivacyAnalyzer is fed with two custom messages in JSON format. The first message was constructed with injecting in the payload an email address as well as names, while the second message is taken from the open Foursquare data set available from <https://sites.google.com/site/yangdingqi/home/foursquare-dataset>.

We have provided the functionalities of the PrivacyAnalyzer Angular User Interface in Deliverable D5.2 [11]. The reader should refer to this deliverable for all information about the information provided by the UI to the end-user. Shortly, PrivacyAnalyzer UI provides the following functionalities to the end-user:

- A report summary that holds aggregate statistics regarding the test session such as the number of critical alerts that have been detected.
- A radar chart that renders the privacy alerts in distinct categories.
- A message logs section. The message log contains important information such as the IPv6 address of the device, the message ID, the class, the level of anonymization and the message topic.

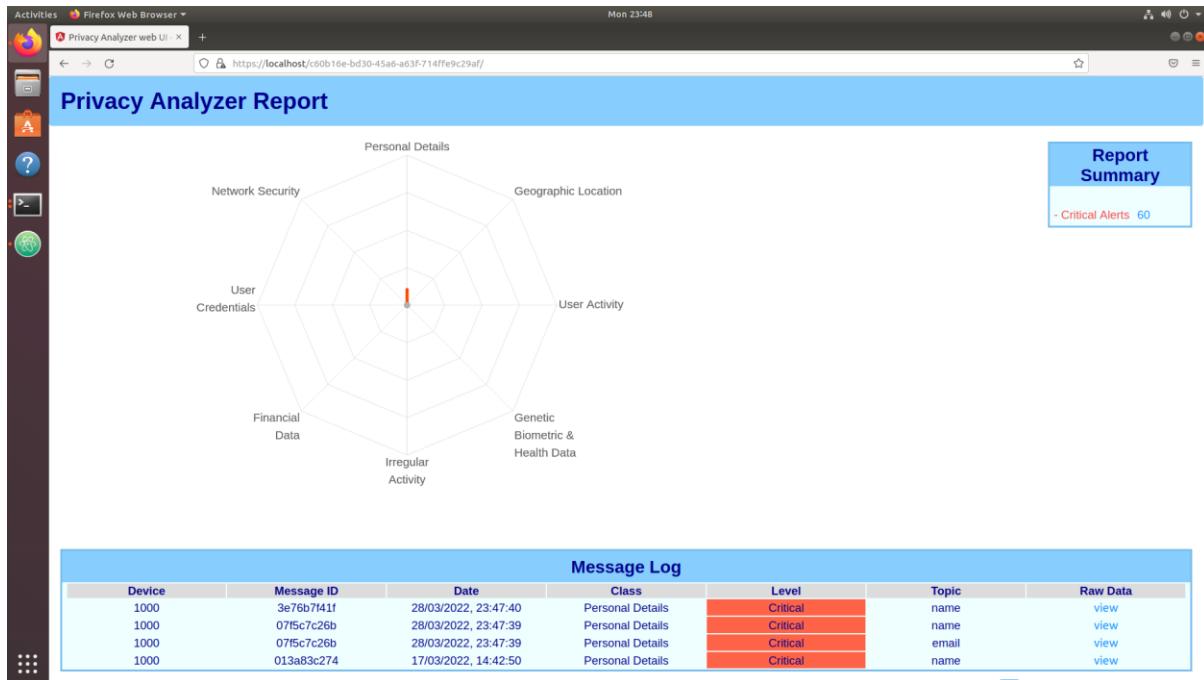


Figure 3.8.3: PrivacyAnalyzer User Interface showing the output of our demo session

As we can see in Figure 3.8.3, for the dummy device with IP address “1000”, the backend detects three critical alerts for the two JSON messages it receives as input. The raw data of each captured message that raised a privacy alert is available to the user through a table at the bottom of the web page as presented in Figure 3.8.4 below.

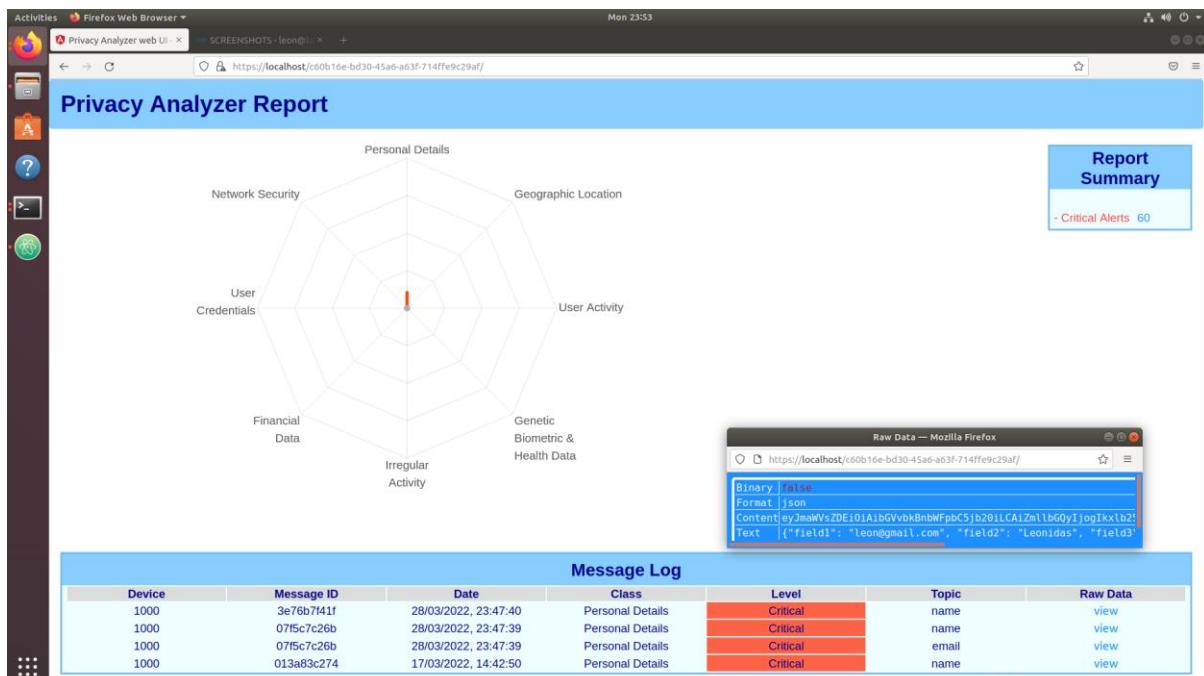


Figure 3.8.4: PrivacyAnalyzer message 1

When we press the ‘view’ link in the third row of the message log, i.e. right to the topic ‘email’, we see that PrivacyAnalyzer detected the pattern “leon@gmail.com” in the payload of the first message. This pattern was classified as an email address and was assigned to the topic

with name “email”. Similarly, the pattern “Leonidas” was detected in the same JSON message and it was classified as a critical alert of type “name”.

The second JSON message from FourSquare message included the ‘checkin’ field with name “Art museum”. Specifically, the used FourSquare message (from the previously discussed data set available from <https://sites.google.com/site/yangdingqi/home/foursquare-dataset>) was the following:

```
{"c1": "470", "c2": "49bb6c0f964a520f4531fe3", "c3": "4bf58dd8d48988d127951735", "c4": "checkins": "Art museum", "c5": "40.719810375488535", "c6": "-74.00258103213994", "c7": "-240", "c8": "Apr 03 18:00:09 +0000 2012"}
```

For the above message, PrivacyAnalyzer found the pattern “Art” as a possible de-identifier and classified this as a critical alert as shown in the left terminal of Figure 3.8.5. Note that the coordinates (in fields “c5” and “c6”) were not detected as our demonstration’s configuration considered GPS coordinates as quasi-identifiers and thus only reported the alerts associated to de-identifiers. Another configuration could treat GPS coordinates as de-identifiers or could include the detected quasi-identifiers in the radar chart and the report and message log sections of our web UI.

The figure displays two terminal windows and a web browser interface. The left terminal shows logs from the PrivacyAnalyzer master node, detailing the detection of personal details and the suppression of sensitive information. The right terminal shows logs from the web service, indicating the successful delivery of anonymized data to the UI. The web browser window shows the 'Privacy Analyzer Report' page, which includes a 'Report Summary' section and a 'Message Log' table listing detected personal details.

Device	Message ID	Date	Class	Level	Topic	Raw Data
1000	3e76b741f	28/03/2022, 23:47:40	Personal Details	Critical	name	view
1000	07f5c7c26b	28/03/2022, 23:47:39	Personal Details	Critical	name	view
1000	07f5c7c26b	28/03/2022, 23:47:39	Personal Details	Critical	email	view
1000	013a83c274	17/03/2022, 14:42:50	Personal Details	Critical	name	view

Figure 3.8.5: PrivacyAnalyzer runtime logs from backend components (left) and web services to UI (right)

Overall, as depicted in Figure 3.8.5, all three detected de-identifiers {"leon@gmail.com", "Leonidas", "Art"} were removed (or suppressed) by the ‘anonymizer’ component of the PrivacyAnalyzer backend.

In the next we shall describe in detail the NS, VNF and Helm Packaging related to the PrivacyAnalyzer Tier-3 service, which we shall employ in the PPDR use case for detecting sensitive information from 5GIOPS UEs (NetApp #9).

### 3.8.2 NetApp analysis

#### 3.8.2.1 NSDs

The PrivacyAnalyzer NetApp one NS at the MEC and one NS at the Core. They are described in the following two tables:

privacyAnalyzer_MECHANICAL_ns	
Number of Services (NSDs)	1
Number of total VNFs	1*
Dependencies of the 5G System	This NetApp shall work in the ININ slice.
How is it operated	K8s deployment in the MEC with a Helm Chart managing Kafka and the docker images {message_receptor, format_detector, pattern_detector, anonym_detector, anonymizer}
Dependencies	Integration API to receive 5G IOPS UE messages by the message_receptor image of PrivacyAnalyzer.

\* as we will discuss in the VNFD section, we have one VNF with a single chart that deploys and configures multiple images

Table 3.8.1: Privacy Analyzer MEC NSD

privacyAnalyzer_CORE_ns	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	This NetApp shall work in the ININ slice.
How is it operated	K8s deployment in the CORE with a Helm Chart managing docker images for MongoDB, web UI and the web services exposed to end-users and administrators.
Dependencies	No.

Table 3.8.2: Privacy Analyzer CORE NSD

#### 3.8.2.2 VNFs

Implementation of Network Services is done by one Virtual Network Function for each Network Service. According to the project's common template for VNF definition, we summarize the requirements of our VNF that are minimum however adequate for the PPDR use case.

Component	PrivacyAnalyzer MEC VNF
Packaging Info	VNF (OSM10)
Internet access	Not mandatory
Placement (latency from-to)	Not mandatory.
Resource req/limits	8 Cores, 16GB Memory, 100GB storage.
Delivery model	Docker containers for K8s

Table 3.8.3: Privacy Analyzer MEC VNF

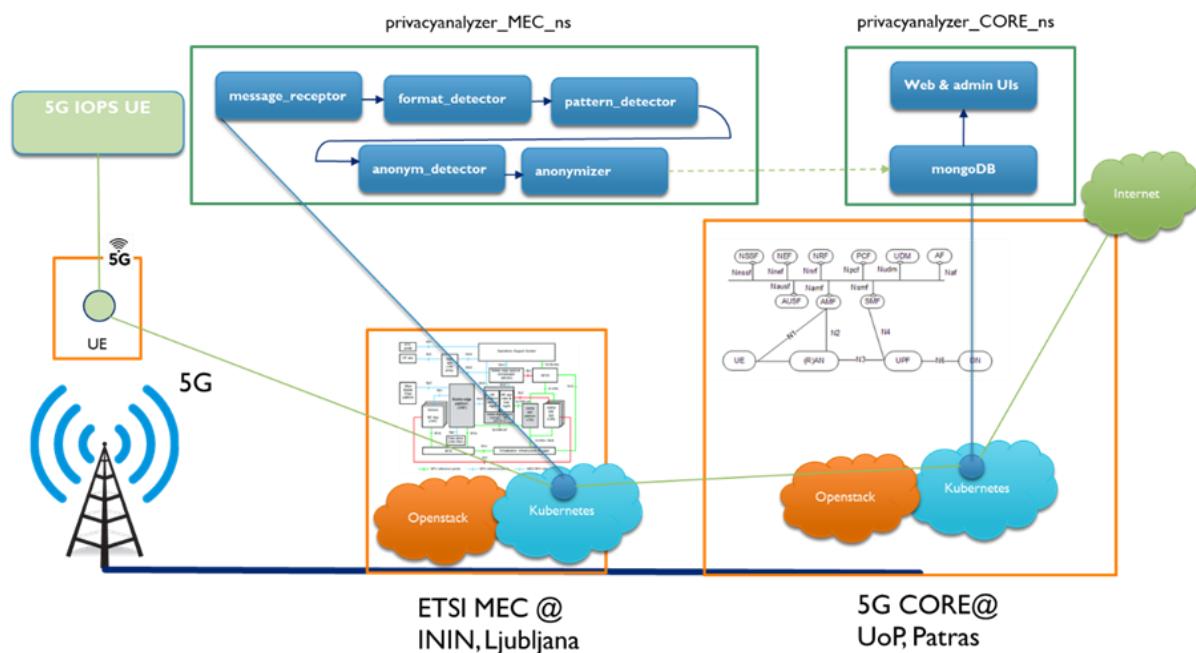
Component	PrivacyAnalyzer CORE VNF
Packaging Info	VNF (OSM10)
Internet access	Yes (for end-users and for admin purposes)
Placement (latency from-to)	Not mandatory.
Resource req/limits	8 Cores, 32GB Memory, 1TB storage (since we hold in MongoDB the history of all analyzed messages; also, more RAM is required for the two-Tiers -frontend and database- of the 3-Tier PrivacyAnalyzer NetApp).
Delivery model	Docker containers for K8s

*Table 3.8.4: Privacy Analyzer CORE VNF*

### 3.8.2.3 NSD and VNF relations/dependencies

For analyzing the messages of 5G IOPS UE, testing sessions will be conducted within the ININ 5G slice that shall be elaborated in the next sub-section (see Section 3.9). PrivacyAnalyzer as discussed at the time of writing of this deliverable shall work in the following way for the PPDR use case, in order to strengthen the privacy of UE's communications with the Edge:

- The ‘privacyanalyzer\_MEC\_ns’ depicted in the Figure 3.8.2 shall receive as input (through an API between Lamda Networks’s code and ININ’s code) JSON messages within the MEC PoP. These shall be processed in order to anonymize possible fields that demand suppression or generalization.
  - The ‘privacyanalyzer\_CORE\_ns’ will be deployed in UoP and it will store all testing session’s messages in a MongoDB cluster. It shall also provide the end-user web User Interface (that we presented earlier) and the UI to the administrator of the NetApp.



*Figure 3.8.6: PrivacyAnalyzer components and data flow*

### 3.8.3 NetApp definition

The definition of PrivacyAnalyzer according to the common template is provided in the following.

NetApp definition:

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Docker images,
  - Helm Charts,
  - NSD/VNFD templates for OSM10.
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: Yes, in port 5000 all docker images respond to the health probes of K8s.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires dynamic both core and MEC deployment in UoP and ININ testbeds respectively.
  - We operate within the slice allocated for ININ's NetApp, thus there is no need for a dedicated slice for PrivacyAnalyzer
- How is it operated (and does it require manual interventions): Configuration of NetApp's images is done via Helm Charts on Harbor for OSM 10.
- Dependencies (does it expose or consume services from/to other NetApps): Our NetApp only consumes data from the 5G IOPS UEs in the same 5G slice.

Therefore, PrivacyAnalyzer's does not have its own NEST per se, but it uses the NEST of the next NetApp, NetApp #9.

#### 3.8.3.1 NSDs definitions

PrivacyAnalyzer's MEC NSD

```
nsd:  
  nsd:  
    - description: NS PrivacyAnalyzer MEC  
      designer: OSM's version, modified by Lamda Networks and tailored for our local OSM with the default K8s  
    vim:  
      df:  
        - id: default-df  
      vnf-profile:  
        - id: privacyanalyzer_MEC_ns_1.0  
          virtual-link-connectivity:  
            - constituent-cpd-id:  
              - constituent-base-element-id: privacyanalyzer_MEC_ns_1.0  
                constituent-cpd-id: mgmt-ext  
                virtual-link-profile-id: mgmtnet  
                vnfd-id: privacyanalyzer_MEC_knf_1.0  
              id: privacyanalyzer_MEC_ns_1.0  
              name: privacyanalyzer_MEC_ns_1.0  
              version: '1.0'  
            virtual-link-desc:  
              - id: mgmtnet  
                mgmt-network: 'true'  
              vnfd-id:  
                - privacyanalyzer_MEC_knf_1.0
```

### PrivacyAnalyzer's CORE NSD

```
nsd:  
  nsd:  
    - description: NS PrivacyAnalyzer CORE  
      designer: OSM's version, modified by Lamda Networks  
      df:  
        - id: default-df  
          vnf-profile:  
            - id: privacyanalyzer_CORE_ns_1.0  
              virtual-link-connectivity:  
                - constituent-cpd-id:  
                  - constituent-base-element-id: privacyanalyzer_CORE_ns_1.0  
                    constituent-cpd-id: mgmt-ext  
                    virtual-link-profile-id: mgmtnet  
                    vnf-id: privacyanalyzer_CORE_knf_1.0  
                  id: privacyanalyzer_CORE_ns_1.0  
                  name: privacyanalyzer_CORE_ns_1.0  
                  version: '1.0'  
                virtual-link-desc:  
                  - id: mgmtnet  
                    mgmt-network: 'true'  
                  vnf-id:  
                    - privacyanalyzer_CORE_knf_1.0
```

### 3.8.3.2 VNFs definitions

#### PrivacyAnalyzer's MEC VNFD

```
vnfd:  
  description: KNF for deployment of Kafka and the PrivacyAnalyzer images message_receptor,  
  format_detector, pattern_detector, anonym_detector, anonymizer, deployed in K8s according to helm chart,  
  named privacyanalyzer_MEC_helm_chart  
  df:  
    - id: default-df  
  ext-cpd:  
    - id: mgmt-ext  
      k8s-cluster-net: mgmtnet  
    id: privacyanalyzer_MEC_knf_1.0  
  k8s-cluster:  
    nets:  
      - id: mgmtnet  
  kdu:  
    - name: privacy_analyzer_service_1.0  
      helm-chart: <lamdanetworks_repo_url>/privacyanalyzer_MEC_helm_chart  
  mgmt-cp: mgmt-ext  
  product-name: privacyanalyzer_MEC_knf_1.0  
  provider: Lamda Networks  
  version: '1.0'
```

#### PrivacyAnalyzer's CORE VNFD

```
vnfd:  
  description: KNF for deployment of images privacyanalyzer_UI, privacyanalyzer_web_service and  
  mongoDB, deployed in K8s according to helm chart, named privacyanalyzer_CORE_helm_chart  
  df:  
    - id: default-df  
  ext-cpd:  
    - id: mgmt-ext  
      k8s-cluster-net: mgmtnet  
    id: privacyanalyzer_CORE_knf_1.0  
  k8s-cluster:  
    nets:  
      - id: mgmtnet  
  kdu:  
    - name: privacy_analyzer_service_1.0  
      helm-chart: <lamdanetworks_repo_url>/privacyanalyzer_CORE_helm_chart
```

```
mgmt-cp: mgmt-ext  
product-name: privacyanalyzer_CORE_knf_1.0  
provider: Lamda Networks  
version: '1.0'
```

In the above KNF descriptions, the MEC K8s deployment is described via the Helm Chart *privacyanalyzer\_helm\_MEC\_chart* which includes all the necessary K8s deployment configurations (e.g., scale out number of Pods, etc) and the K8s health checks. Similarly, the CORE K8s deployment is described in the Helm chart *privacyanalyzer\_helm\_CORE\_chart*.

Note that the name of our private Harbor helm chart repository hosted in Google cloud is in our VM with IP address “*lamdanetworks\_repo\_url*”. It shall be changed to the 5GASP Harbor URL once available.

### 3.8.4 NetApp’s tests plan and definition

For functional testing of the NetApp, we will use the data sets emulating network messages which we use for demonstrations of PrivacyAnalyzer. This data set is a synthesis of publicly available open-source data, which have sensitive information in terms of de-identifiers and quasi-identifiers which we have discussed in detail in section 3.8.A of deliverable D5.2 (edited by Lamda Networks) [11]. For example, we have Foursquare data sets that contain GPS coordinates, and these are quasi-identifiers, i.e., they can lead to Personal Identifiable Information (PII) disclosure. We also use other data sets such as open data from repositories like <https://openaddresses.io>.

Functional testing is performed by measuring the accurate detection of privacy alerts when PrivacyAnalyzer’s “message\_receptor” component is fed with JSON-formatted entries constructed from our test data sets (we construct these entries with python custom modules tailored to the specific data sets that we use; for other types of data sets we need to write some extra custom modules). As an example, if we use data set X known to have  $n_1$  “Critical Alerts of Personal Information” and  $n_2$  “Critical Alerts of Geographic Information”, then we can derive that our software is functional when in the Web UI we obtain that the results of the privacy analysis results are:  $n_1$  “Critical Alerts of Personal Information” and  $n_2$  “Critical Alerts of Geographic Information”. The reasons that may lead the functional tests to fail are associated to loss of messages which we observed when testing our software with real IoT devices. We could observe the same failures in the PPDR use case when we proceed with the testing sessions of 5G IOPS UEs (see Section 3.9 NetApp #9).

Performance testing is measured by means of the time taken to process a certain number of JSONs, which we configure in our tests. We consider that the performance test is met when we process 100K messages in less than one second with a speedup factor of 1 in the non-Spark deployment of the software. The payload of each message is 1Kbyte. When we employ Spark to speedup message processing (which is typically used in cloud deployments, not in the MEC deployment that we shall conduct for the PPDR case in this project) then we consider the performance test as passed if we obtain processing of 400-500K messages with payload 1Kbyte each message in a time under 1 second.

Finally, we shall use the tests defined in WP5 D5.3 [3]. Specifically, we plan to conduct the following tests:

- mandatory tests: 4
- optional tests: 6, 9, 14

### 3.8.5 NetApp requirements for testbeds

In terms of slice requirements, PrivacyAnalyzer will use the PPDR slice which shall be described in the next subsection describing NetApp #9.

**Hardware requirements:** PrivacyAnalyzer is employed in the PPDR use case. In the latter, an API for exchanging messages has been discussed with partner ININ. The purpose of this API is that selected 5G IOPS UEs send JSON messages to PrivacyAnalyzer's relevant endpoint. The status is ongoing in respect to the schema of the messages based on the hardware of ININ. Finally, the requirements (CPU, RAM, HD) for hosting our K8s deployment is described in sub-section "VNFs" 3.8.2.2.

**Software requirements:** OSM 10 with a K8s cluster is needed for both the MEC NS and the CORE NS of PrivacyAnalyzer, which have been described in sub-section 3.8.2.1.

## 3.9. NetApp #9

### 3.9.1 NetApp functional overview, current status and next steps

A goal of the “Isolated Operations for Public Safety” NetApp is to maintain communication capabilities between public safety users in the field, offering them local mission critical services, even in a case the backhaul connectivity to the core network is not fully functional or is disrupted. Such an operational mode is typically needed in PPDR disaster situations, when the infrastructure is damaged or destroyed, and in the out of coverage emergency cases operated in remote areas. Based on a user requirement, as already described, the NetApp needs to be provisioned via two deployment scenarios – standalone (the backhaul connectivity to the core network is not fully functional or is disrupted, see also Figure 3.9.1) and distributed (complete network works without any disruption see also Figure 3.9.2). Standalone deployment represents the basic deployment scenario where both NetApp components, i.e., 5G core function (Mobile Core VNF) and RAN function (Cloud Baseband Unit VNF), are placed in the edge to provide isolated 5G services to PPDR users. On the other hand, the distributed deployment scenario supports separating 5G core function from the RAN function on a network function level.

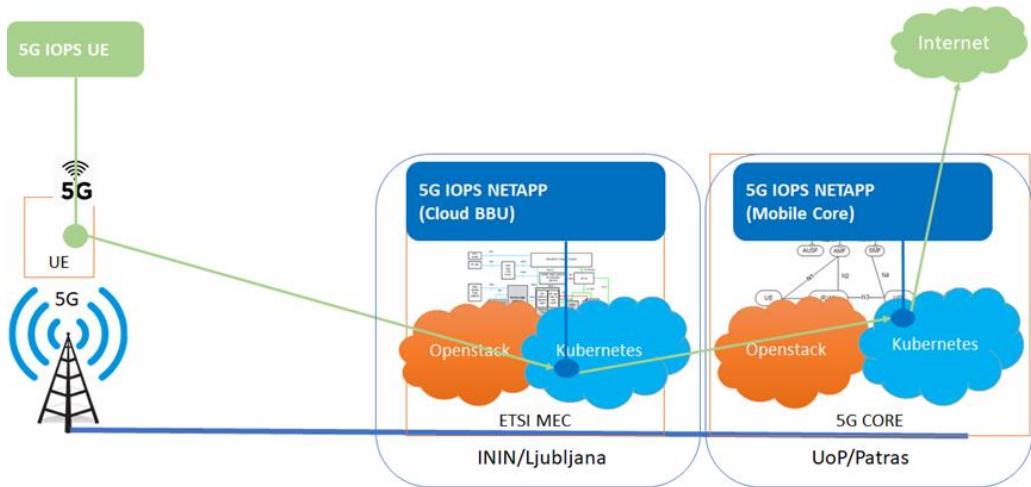


Figure 3.9.1: Regular day-to-day operation phase, where the NetApp is deployed in the distributed manner, internet access and 5G services are provided to PDDR users from the public 5G core network (although Mobile Core VNF is deployed at the MEC, it is not active in this case)

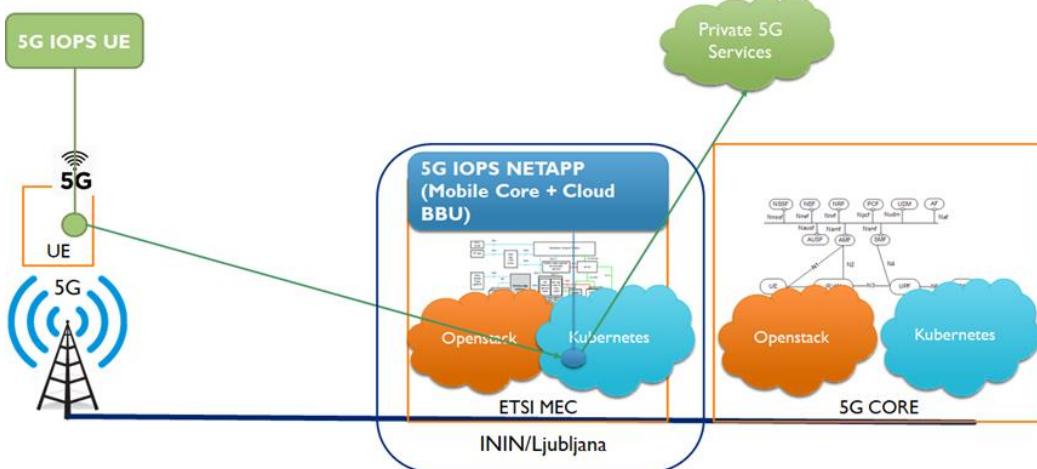


Figure 3.9.2: Isolated operations mode following uplink failure detection (based on the health-check mechanism implemented) and the reconfiguration of the Cloud BBU VNF to use the Mobile core VNF provisioned on the edge/MEC, enabling basic/private 5G services to the users

Docker images for the NetApp components have been prepared and packaged as VM/VNF (VNF and based orchestration):

- gNB - Cloud BBU VNF,
- CN - Mobile core VNF,
- helpers (license, UI).

Figure 3.9.3 presents gNB and CN VNFs docker image creation and run, while Figure 3.9.4 shows status of the same processes, i.e., being up and running.

```
sudo docker run -tid \
--name ltemme \
--mount type=bind,source="$(pwd)"/ltemme/config/mme.template.cfg,target=/app/mme/config/mme.template.cfg \
--mount type=bind,source="$(pwd)"/ltemme/config/ue_db-ims.cfg,target=/app/mme/config/ue_db-ims.cfg \
--mount type=bind,source="$(pwd)"/ltemme/ltemme_service_logs,target=/tmp \
--mount type=bind,source="$(pwd)"/ltemme/logs,target=/app/screen_logs \
--net=host \
--device /dev/net/tun:/dev/net/tun \
--cap-add=NET_ADMIN \
--env-file lte.conf \
core-harbor.qmon.eu/5gsystem/ltemme:$RADIO_TYPE-$SOFTWARE_VERSION

sudo docker run -dit \
--name lteenb \
--restart=always \
--mount type=bind,source="$(pwd)"/lteenb/config/enb.template.cfg,target=/app/enb/config/enb.template.cfg \
--mount type=bind,source="$(pwd)"/lteenb/logs,target=/app/screen_logs \
--mount type=bind,source="$(pwd)"/lteenb/lteenb_service_logs,target=/tmp \
--net=amarisoft --ip 172.18.0.3 \
--cap-add=SYS_RAWIO -v /dev:/dev --privileged \
--env-file lte.conf \
-p 9002:9000 \
core-harbor.qmon.eu/5gsystem/lteenb:sdr-$SOFTWARE_VERSION
```

Figure 3.9.3: Using “`docker run`” command to create a writeable container layer over the specified images of CN and gNB, and starting them

*Figure 3.9.4: Required NetApp processes are up and running*

Configuration parameters exposure through ENV parameters at container level, enabling day 0 – 2 configurations, has been done enabling fully configurable 5G system parameters, e.g., MCC/MNC, ARFCN, AMF address, GTP address, etc. The following list shows configurable parameters and their default values set at the development phase:

```
- name: license-server-address # mandatory - configure license server address
  value: 0.0.0.0
- name: plmn # mandatory - define PLMN
  value: "00101"
- name: operation-mode # mandatory - define operation mode (1 - gNB, 2 - gNB+CN, 3 - CN)
  value: 2
  data-type: INTEGER
- name: slice-type # mandatory - define network slice type (embb)
  value: embb
- name: iops-operation # optional - if IOPS operation is enabled
  value: True
  data-type: BOOLEAN
- name: iops-public-core-plmn # optional - define remote core PLMN (mandatory if IOPS operation is True)
  value: "00102"
- name: iops-public-core-amf-address # optional - define remote core AMF address (mandatory if IOPS operation is True)
  value: 0.0.0.0
- name: iops-public-core-healthcheck-type # optional - define healthcheck type (1 - ICMP)
  value: 1
  data-type: INTEGER
- name: iops-public-core-healthcheck-interval # optional - define healthcheck interval in seconds
  value: 30
  data-type: INTEGER
- name: iops-public-core-healthcheck-threshold # optional - define healthcheck threshold for failed/success attempts
```

value: 3
data-type: INTEGER

In Figure 3.9.5, showing ENV sample file, it can be observed three parts of the file, important from the functional point of view:

- default 5G IOPS PLMN-Id,
- parameters of 5G IOPS core and
- parameters of 5G IOPS gNB followed by parameters of the remote (public) core, which is 5G IOPS gNB connected to, during day-to-day operations scenario (distributed mode).

```

# [GENERAL]
# Version of the Amarisoft LTE Software
SOFTWARE_VERSION=2021-09-17
SWALLOW_VERSION=6.11
RADIO_TYPE=rrh
# PLMN identity of the MME (5 or 6 digits).
PLMN=00101 5G IOPS PLMN definition

# [LICENSE]
# Configuration of the Amarisoft license server to use. String. IP address of the license server.
LICENSE_SERVER_ADDRESS=192.168.202.82

# [LOGGING]
# Rename current log every time it reaches size bytes open new one. Size is an integer and can be followed by K, M or G.
LOG_FILE_MAX_SIZE=1G
# Number of logs files that are not deleted. When the threshold is reached, the older files are automatically deleted.
LOG_MAX_NUBMER_OF_FILES=10

# [MME]
GTP_ADDRESS_MME=127.0.1.100
GTP_SUBNET=127.0.1.0/16
COM_ADDRESS_MME=0.0.0.0:9000
UPLINK_INTERFACE=enp7s0

# [ENB]
#GTP_ADDRESS_ENB=127.0.1.1
#COM_ADDRESS_ENB=0.0.0.0:9001
#AMF_ADDR=127.0.1.100
GTP_ADDRESS_ENB=172.20.1.10
COM_ADDRESS_ENB=0.0.0.0:9001
AMF_ADDR=192.168.202.43
  
```

Figure 3.9.5: ENV sample file

Initial development tests have been performed for the current implementation of the system, i.e., VM/Docker-based 5G IOPS system:

- 5G IOPS standalone mode (both Cloud BBU and Mobile core VNFs active on MEC),
- 5G distributed mode (Cloud BBU VNF is active on MEC, Mobile core VNF is active in 5G core).

Technically speaking, 5G IOPS standalone mode is activated after expiration of *iops-public-core-healthcheck-interval*(=30 s) for *iops-public-core-healthcheck-threshold-times* (=3 times), based on healthcheck method defined by *iops-public-core-healthcheck-type* (=response to ICMP packets) – see also the above list of ENV parameters. Switching cores is further depicted in Figure 3.9.6. In Grafana dashboard screenshots (Figure 3.9.7 and Figure 3.9.8), we can observe additional characteristics when in 5G IOPS mode:

- radio settings of the 5G UE connected to 5G IOPS slice (Figure 3.9.7),
- average download speed of the 5G UE connected to 5G IOPS slice (Figure 3.9.8).

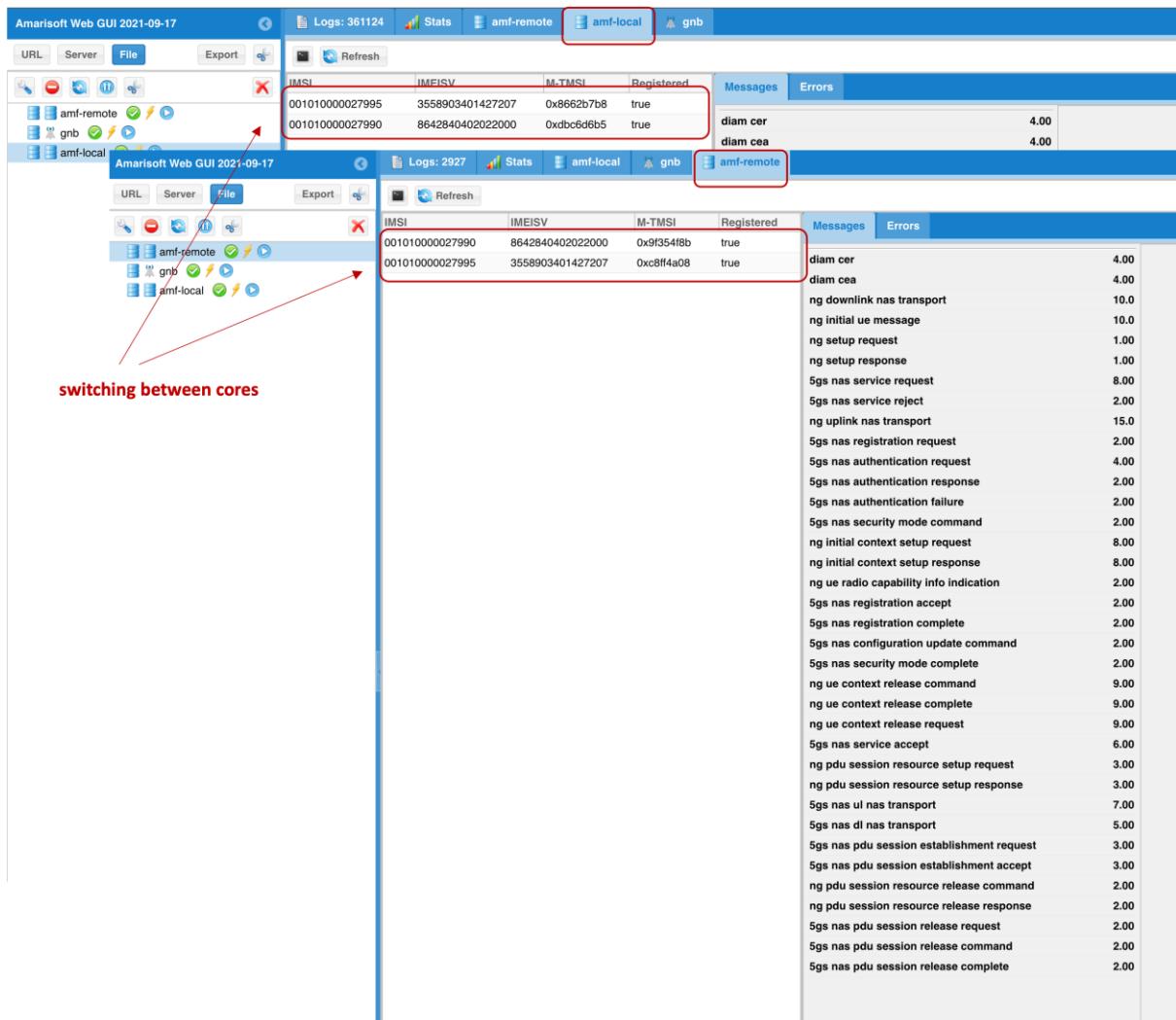


Figure 3.9.6: Details on switching between the two cores, i.e., switching from distributed to standalone mode

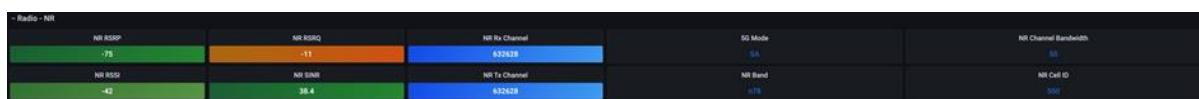


Figure 3.9.7: Grafana dashboard showing radio settings for the 5G UE connected to 5G IOPS slice

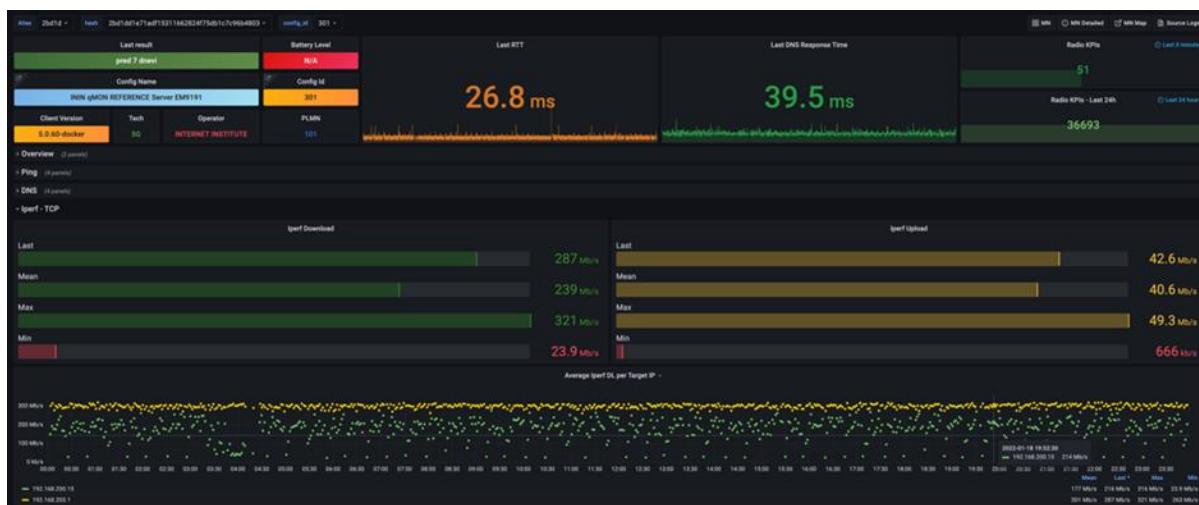


Figure 3.9.8: Grafana dashboard showing average download speed for the 5G UE connected to 5G IOPS slice

Temporarily, end-to-end testing of the 5G IOPS system operation is in progress, focusing on the following tasks:

- dual-SIM operation: supporting continuous operation of the UE while switching between distributed and standalone mode of operation,
- gNB-AMF keepalive: detecting disruptions/failures on the backhaul connection, i.e., connectivity between physical locations of MEC and 5G core,
- performance and stability tests.

In relation to the NetApp design, implementation of the first version of the VNFD and NSD descriptors, and development of VNF/NS packaging and onboarding to OSM 10 in the Ljubljana/ININ testbed has been done (see Figure 3.9.10, Figure 3.9.11, Figure 3.9.12). Testing via CI/CD pipeline will follow, as well as further versions of the NetApp will be deployed in a similar manner.

Instances

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
SGASPIOPS-GNB-0-1->OSYSTEM_ONB_vnfd-VM-0	5G-SYSTEM-VNF	172.20.3.49	4-4-80-ef01	-	Active	slice	None	Running	17 hours, 48 minutes	Create Snapshot
SGASPIOPS-REMOT-0->OSYSTEM_CN_vnfd-VM-0	5G-SYSTEM-VNF	172.20.3.97	2-4-80	-	Active	none	None	Running	18 hours, 17 minutes	Create Snapshot

Figure 3.9.9: Deployment of CN VNF/VM and IOPS VNF/VN in OSM10.

Name	Identifier	Host Name	Operational Status	Config Status	Default Status	Actions
SGASPIOPS-GNB-05Mv3	8a8f80d2-25ce-4674-82ee-171c179e0d5f	v5GSYSTEM_GNB_mnf	green	green	Done	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Actions</a>
SGASPIOPS-REMOT-CN-05Mv3	3294b4a4-4c23-4c22-823a-677083c0e8b	v5GSYSTEM_CN_mnf	green	green	Done	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Actions</a>

Figure 3.9.10: Deployment of CN NS and IOPS NS in OSM10

ID	Action	Output	Status	Verified	Code
2	verify-ssh-credentials	-	completed	True	0
4	configure-service	-	completed	-	0
6	start-service	Starting services... f2845da7f0de047839bc53899433485f3d1a677c38e3400e8647f9577a51e 4411974abeb5e0f223f672220bd7c71fc16a82825fe8282520f5a181f58 d7b0f148847v214809a50c783d703138b27d721182e552610f51ae407e799	completed	-	0

Figure 3.9.11: Day-1 configuration for the 5G IOPS service in OSM10

## Actions

Actions	Description
configure-service	Configures the 5g service
generate-ssh-key	Generate a new SSH keypair for this unit. This will replace any existing previously generated keypair.
get-ssh-public-key	Get the public SSH key for this unit.
reconfigure-iops	Reconfigure IOPS service
reconfigure-iops-healthcheck	Reconfigure IOPS healthcheck
restart-service	Restarts the 5G service of the VNF
run	Run an arbitrary command
start-iops-service	Starts the IOPS service
start-service	Starts the 5G service of the VNF
stop-iops-service	Stops the IOPS service
stop-service	Stops the 5G service of the VNF
verify-ssh-credentials	Verify that this unit can authenticate with server specified by ssh-hostname and ssh-username.

Figure 3.9.12: Actions/commands available in day-1/day-2 configuration of the 5G IOPS NetApp

### 3.9.2 NetApp analysis

#### 3.9.2.1 NSDs

The NetApp uses three Network Services, described by their respective Network Service Descriptors (NSDs). Network services functionalities are presented in the previous sections. Network Services:

- BBU,
- CN,
- IOPS: combined Network Service consists of BBU and CN.

BBU NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	<ul style="list-style-type: none"> <li>• Dynamic VM deployment and MANO connectivity to recover deployment status information</li> <li>• “core” slice</li> </ul>
How is it operated	Using proxy charms
Dependencies	No

Table 3.9.1: BBU NSD

CN NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information

How is it operated	Using proxy charms
Dependencies	No

Table 3.9.2: CN NSD

IOPS (CN+BBU) NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information
How is it operated	Using proxy charms
Dependencies	No

Table 3.9.3: IOPS (CN+BBU) NSD

### 3.9.2.2 VNFs

Implementation of Network Services is done by Virtual Network Functions which are defined in Table 3.9.4, Table 3.9.5 and Table 3.9.6. Three VNFs have been created for the 5G IOPS NetApp to implement Network Services defined in section 3.9.2.1:

- 5G IOPS BBU,
- 5G IOPS Mobile Core,
- 5G IOPS CN+BBU.

Component	5G IOPS BBU NS
Packaging Info	VNF (OSM10)
Internet access	No
Placement (latency from-to)	Latency from UE: max 10ms
Resource req/limits	4 Cores, 16GB Memory, 20GB storage
Delivery model	VM (Containers running inside) VNF/proxy charms

Table 3.9.4: 5G IOPS BBU NS

Component	5G IOPS Mobile Core NS
Packaging Info	VNF (OSM10)
Internet access	Yes
Placement (latency from-to)	Latency from UE: max 10ms
Resource req/limits	4 Cores, 16GB Memory, 20GB storage
Delivery model	VM (Containers running inside) VNF/proxy charms

Table 3.9.5: 5G IOPS Mobile Core NS

Component	5G IOPS CN+BBU NS
Packaging Info	VNF (OSM10)
Internet access	Yes
Placement (latency from-to)	Latency from UE: max 10ms
Resource req/limits	4 Cores, 16GB Memory, 20GB storage
Delivery model	VM (Containers running inside) VNF/proxy charms

Table 3.9.6: 5G IOPS CN+BBU NS

### 3.9.2.3 NSD and VNF relations/dependencies

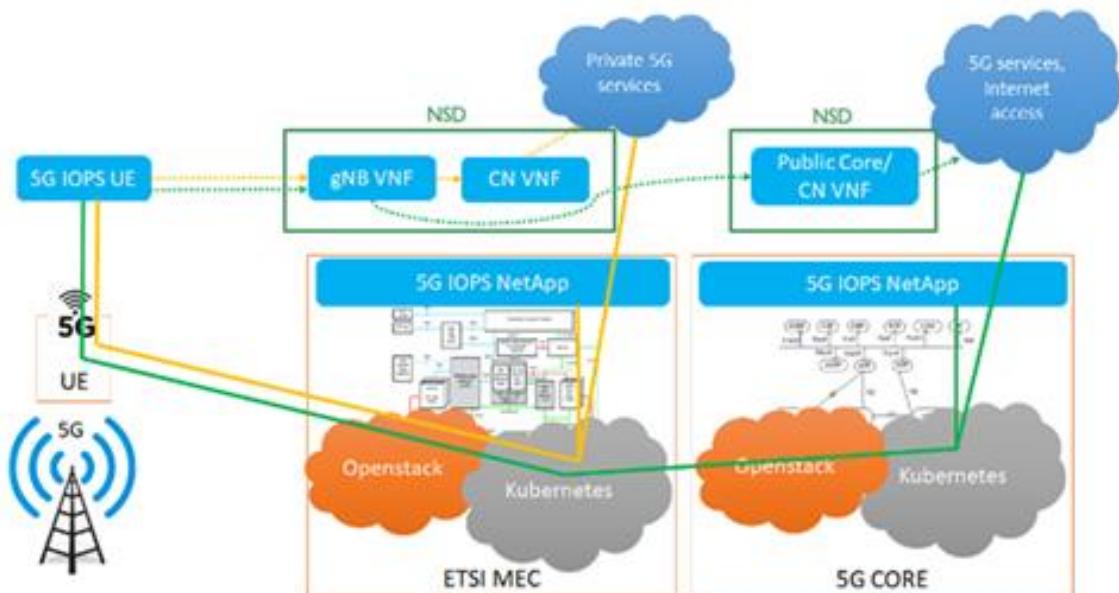


Figure 3.9.13: Relations between NSDs and VNFs.

For the NetApp deployment within the testbed infrastructure, enabling end-to-end use case showcasing, relations and dependencies between NSDs and VNFs need to be considered as well. The realization of the use case is planned as follows:

- public network core emulation (distributed or day-to-day mode of operation): single VNF (5G IOPS Mobile Core) used to realize required CN (Core Network) network service,
- IOPS node emulation (MEC node): single VNF (5G IOPS CN+BBU) for the realization of the IOPS network service, i.e., “IOPS (CN+BBU)” as defined in section 3.9.2.1, however, there were two implementation possibilities identified of which the first one is preferred at the moment:
  - using single VDU with multiple containers (i.e., gNB and CN),
  - using two VDUs each running single container (i.e., gNB or CN).

### 3.9.3 NetApp definition

NetApp definition:

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Virtual machine with custom image,
  - custom flavor,
  - NSD/VNFD templates (OSM10).
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: No.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires dynamic VM deployment and MANO connectivity to recover deployment status information.
  - Since the IOPS NetApp deploys the whole slice, for the IOPS scenario only core functions could be provisioned via initial slice deployment (TBD).
- How it is operated (and if it requires manual interventions): Configuration of NetApp components (gNB,CN) is done via proxy charms (OSM 10).
- Dependencies (does it expose or consume services from/to other NetApps?): No.

5G IOPS NETAPP's NEST	
Area of service	SI
Area of service: Region specification	Ljubljana
Isolation level	Virtual resources isolation
Mission critical support	1: mission-critical
Mission critical support: Mission-critical capability support	1: inter-user prioritization
Mission critical support: Mission-critical service support	2: MCDATA
Mission critical support: Mission-critical service support	3: MCVideo
Slice quality of service parameters: 3GPP 5QI	69, 70
Maximum Packet Loss Rate	10^-6
Supported device velocity	0-6km/h

Table 3.9.7: 5G IOPS NetApp's NEST

#### 3.9.3.1 NSDs definitions

CN+BBU NSD:

```

nsd:
  nsd:
    - description: Generated by ININ
      designer: ININ
    df:
      - id: default-df
        vnf-profile:
          - id: '1'
            virtual-link-connectivity:
              - constituent-cpd-id:
                  - constituent-base-element-id: '1'
                    constituent-cpd-id: vnf-cp0-ext

```

```

virtual-link-profile-id: 5GASP-flat
vnfd-id: v5GSYSTEM_IOPS_vnfd
id: v5GSYSTEM_IOPS_nsd
name: v5GSYSTEM_IOPS_nsd
version: '1.0'
virtual-link-desc:
- id: 5GASP-flat
  mgmt-network: true
vnfd-id:
- v5GSYSTEM_IOPS_vnfd

```

### 3.9.3.2 VNFs definitions

#### CN+BBU VNFD:

```

vnfd:
description: Generated by ININ
df:
- id: default-df
instantiation-level:
- id: default-instantiation-level
vdu-level:
- number-of-instances: 1
  vdu-id: v5GSYSTEM_IOPS_vnfd-VM
vdu-profile:
- id: v5GSYSTEM_IOPS_vnfd-VM
  min-number-of-instances: 1
lcm-operations-configuration:
operate-vnf-op-config:
day1-2:
- id: v5GSYSTEM_IOPS_vnfd-VM
execution-environment-list:
- id: configure-vnf
  external-connection-point-ref: vnf-cp0-ext
juju:
charm: charm
initial-config-primitive:
- execution-environment-ref: configure-vnf
  name: config
  parameter:
  - name: ssh-hostname
    value: <rw_mgmt_ip>
  - name: ssh-username
    value: ubuntu
  - name: ssh-password
    value: qoe
  seq: '1'
- name: configure-service
  seq: '2'
  parameter:
  - name: license-server-address # mandatory - configure license server address
    value: 0.0.0.0
  - name: plmn # mandatory - define PLMN
    value: "00101"
  - name: operation-mode # mandatory - define operation mode (1 - gNB, 2 - gNB+CN, 3 - CN)
    value: 2
    data-type: INTEGER
  - name: slice-type # mandatory - define network slice type (embb)
    value: embb
  - name: amf-address # optional - if operation-mode=1 remote core can be used by default, define AMF
address here

```

```

    value: 0.0.0.0
    - name: radio-network-name # optional - radio network name
      value: "INTERNET INSTITUTE"
    - name: radio-short-network-name # optional - radio network short name
      value: "ININ"
    - name: radio-tdd # optional - if radio TTD mode is enabled
      value: True
      data-type: BOOLEAN
    - name: radio-arfcn # optional - define radio ARFCN
      value: "632628"
    - name: radio-bw-mhz # optional - define channel bandwidth in Mhz
      value: 50
      data-type: INTEGER
    - name: radio-transport-profile # optional - define transport profile (1,2,3)
      value: 2
      data-type: INTEGER
    - name: radio-sdr-rx-gain # optional - if SDR is used
      value: 55
      data-type: INTEGER
    - name: radio-sdr-tx-gain # optional - if SDR is used
      value: 90
      data-type: INTEGER
    - name: iops-operation # optional - if IOPS operation is enabled
      value: True
      data-type: BOOLEAN
    - name: iops-public-core-plmn # optional - define remote core PLMN (mandatory if IOPS operation is
True)
      value: "00102"
    - name: iops-public-core-amf-address # optional - define remote core AMF address (mandatory if IOPS
operation is True)
      value: 0.0.0.0
    - name: iops-public-core-healthcheck-type # optional - define healthcheck type (1 - ICMP)
      value: 1
      data-type: INTEGER
    - name: iops-public-core-healthcheck-interval # optional - define healthcheck interval in seconds
      value: 30
      data-type: INTEGER
    - name: iops-public-core-healthcheck-threshold # optional - define healthcheck threshold for
failed/success attempts
      value: 3
      data-type: INTEGER
    - name: start-service
      seq: '3'
config-primitive:
  - execution-environment-ref: configure-vnf
    name: restart-service
  - name: stop-service
  - name: stop-iops-service
  - name: start-iops-service
  - name: reconfigure-iops
parameter:
  - name: iops-public-core-plmn
    default-value: ""
    data-type: STRING
  - name: iops-public-core-amf-address
    default-value: ""
    data-type: STRING
  - name: reconfigure-iops-healthcheck
parameter:
  - name: iops-public-core-healthcheck-type
    default-value: ""

```

```

    data-type: INTEGER
    - name: iops-public-core-healthcheck-interval
    default-value: ""
    data-type: INTEGER
    - name: iops-public-core-healthcheck-threshold
    default-value: ""
    data-type: INTEGER

ext-cpd:
- id: vnf-cp0-ext

int-cpd:
    cpd: ens3-int
    vdu-id: v5GSYSTEM_IOPS_vnfd-VM
id: v5GSYSTEM_IOPS_vnfd
mgmt-cp: vnf-cp0-ext
product-name: v5GSYSTEM_IOPS_vnfd
provider: ININ
sw-image-desc:
- id: 5G-SYSTEM-VNF
  image: 5G-SYSTEM-VNF
  name: 5G-SYSTEM-VNF
vdu:
- description: v5GSYSTEM_IOPS_vnfd-VM
  id: v5GSYSTEM_IOPS_vnfd-VM
  int-cpd:
    - id: ens3-int
      virtual-network-interface-requirement:
        - name: ens3
          virtual-interface:
            type: PARAVIRT
  name: v5GSYSTEM_IOPS_vnfd-VM
  sw-image-desc: 5G-SYSTEM-VNF
  virtual-compute-desc: v5GSYSTEM_IOPS_vnfd-VM-compute
  virtual-storage-desc:
    - v5GSYSTEM_IOPS_vnfd-VM-storage
  monitoring-parameter:
    - id: vnf_cpu_util
      name: vnf_cpu_util
      performance-metric: cpu_utilization
    - id: vnf_memory_util
      name: vnf_memory_util
      performance-metric: average_memory_utilization
    - id: vnf_packets_sent
      name: vnf_packets_sent
      performance-metric: packets_sent
    - id: vnf_packets_received
      name: vnf_packets_received
      performance-metric: packets_received
version: '1.0'
virtual-compute-desc:
- id: v5GSYSTEM_IOPS_vnfd-VM-compute
  virtual-cpu:
    num-virtual-cpu: 4
  virtual-memory:
    size: 4.0
  virtual-storage-desc:
    - id: v5GSYSTEM_IOPS_vnfd-VM-storage
      size-of-storage: 80

```

### 3.9.4 NetApp's tests plan and definition

The NetApp will follow common test plan defined in WP5 [3] with selected test cases to be performed:

- mandatory tests: 1, 3, 4, 19
- optional tests: 2, 6, 12, 13, 15

Alongside, for the IOPS NetApp, the end-to-end network performance testing will be implemented based on ININ's qMON Monitoring Tool. Main components of qMON Monitoring System are the following:

- **qMON Network Sensor** – Autonomous and distributed agents for user and services emulation,
- **qMON Reference Server** - Reference measurement endpoint for qMON NetworkSensor components,
- **qMON Manage** – Central cloud-based agent management and monitoring,
- **qMON Collector** – Central data collection and enrichment infrastructure,
- **qMON Insight** – Online and offline analytics for real-time status monitor and advanced root-cause analysis.

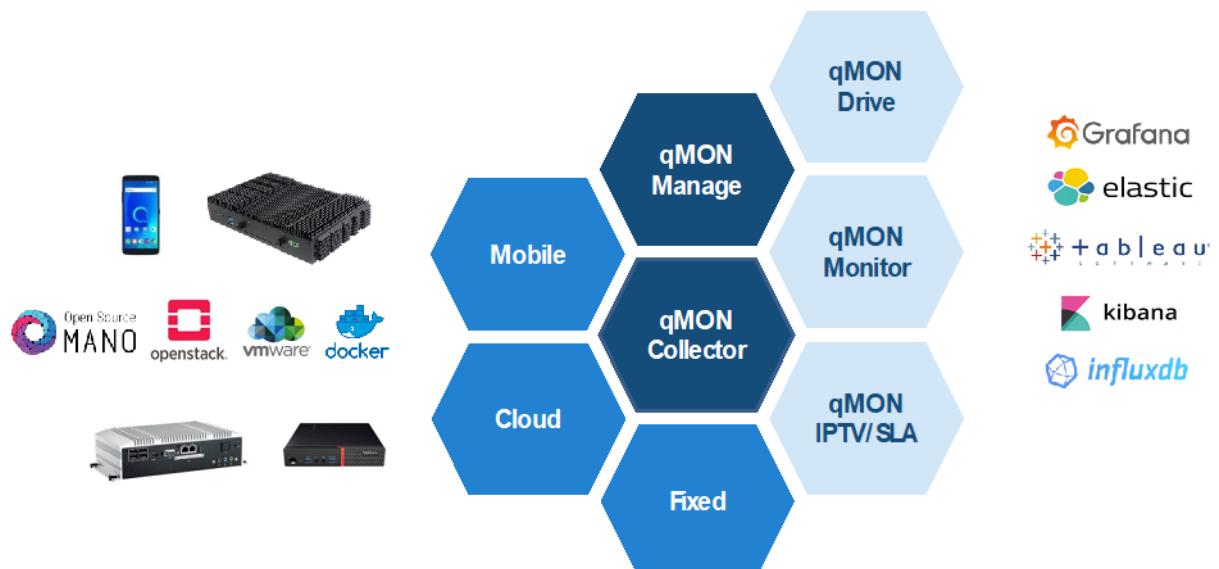


Figure 3.9.14: qMON Monitoring Tool System Modules

The planned test environment is as follows:

- 5G UE
  - implemented as an industrial PC with integrated 5G modem (Sierra Wireless EM9191)
  - capable of running and orchestrating containers via Kubernetes platform (based on k3s locally installed on the 5G UE)
  - **qMON Network Sensor** component deployed as a helm chart
- 5G IOPS components deployed via OSM10
  - each VNF that involves deploying CN containers also deploys a **qMON Reference Server** container to provide measurement endpoints for 5G UE

running qMON NetworkSensors – this allows creating end-to-end tests to CN independently from where it is deployed (i.e., IOPS or public cloud)

- optional: **qMON Reference Server** predeployed in different network segments
- **qMON Collector** deployed and available (ININ cloud, not part of dynamic instantiation)
- **qMON Manage** deployed and available (ININ cloud, not part of dynamic instantiation)
- **qMON Insight** deployed and available (ININ cloud, not part of dynamic instantiation)

The test case will follow these steps:

- deploy CN NSD/VNF to serve as a “remote” core to which IOPS NetApp connects
  - run **qMON Reference Server** container
- deploy IOPS NetApp which in turn instantiates IOPS slice and attaches to “remote” CN
- configure **qMON Manage** to assign proper work orders for the qMON NetworkSensor (API invoking for predefined work orders is possible, which allows certain automation steps)
- deploy 5G UE
  - perform radio and network attachment (automation could be implemented by rebooting UE which triggers reconnect)
  - run **qMON Network Sensor** container
- emulate connectivity problems between IOPS NetApp and “remote” core
- IOPS NetApp switches to using local IOPS core
  - Run **qMON Reference Server** container (on local IOPS core)
- 5G UE
  - perform radio and network attachment (automation could be implemented by rebooting UE which triggers reconnect)
- results are gathered in **qMON Collector** and can be extracted to use with other tools (via MySQL) or can be observed via templated Grafana dashboard on **qMON Insight**

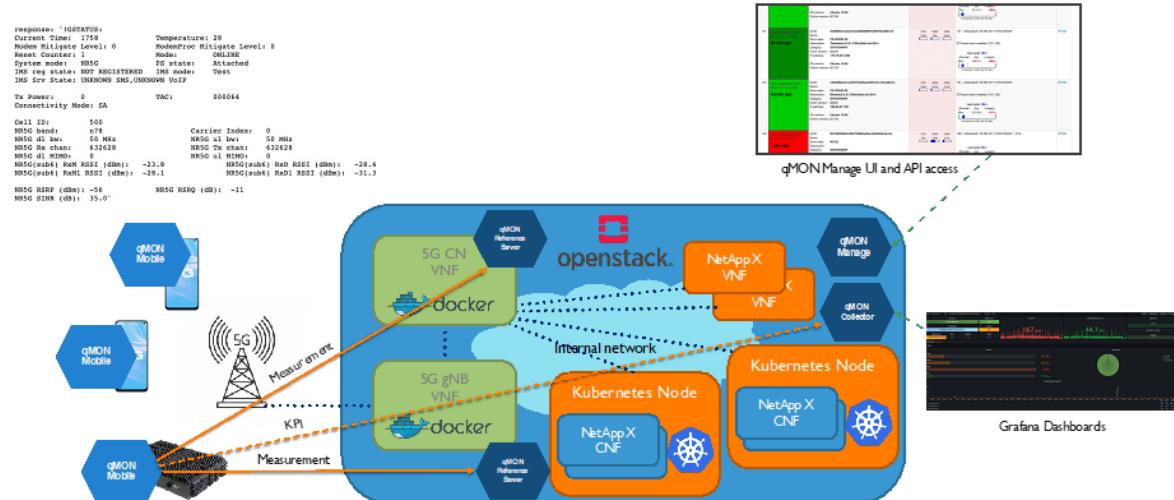


Figure 3.9.15: Integration of end-to-end NetApp performance testing in ININ's testbed using qMON Monitoring Tool

```

2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - -----GPS MEASUREMENT-----
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - latitude: 46.0358883333
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - lon_direction: E
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - lat_direction: N
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - num_sats: 08
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - duration: 0.163713932037
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - timestamp: 120151.200
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - altitude: 305.8
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - altitude_units: M
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - longitude: 14.4778583333
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - -----GPS MEASUREMENT-----
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - VDOP: 1.6
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - PDOP: 1.8
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - HDOP: 0.9
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - -----GPS MEASUREMENT-----
2022-03-03 12:01:51 - INFO - LOG.Log.LogFile - speed over ground knots: 0.17
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - -----iperf measurement-----
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Server: 192.168.203.1
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Protocol: TCP
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Direction: DL
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Duration: 10
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Timeout: 25
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Bandwidth_kbps: 10000
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - TCP MSS: 1260 bytes
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Used port: 10000
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Number of parallel streams: 5
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Status: Success
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Results: {'tcp_mss': 1260, 'transfer_bytes': 248696864, 'num_streams': 5, 'retransmits': 635, 'duration': 10.000171, 'kbps': 198954.09, 'tcp_window_size': '-1'}
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - -----traceroute-----
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Route: ['192.168.203.1']
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - First hop RTT: -1
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - -----radio measurement-----
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Operator name: INTERNET INSTITUTE
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Operator code: 00101
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Access type: 5G
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Registration state: 1
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - Cell ID: 500
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_tx_power_dbm: -24
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_rsrp_dbm: -90
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_carrier_index: 0
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_ul_mimo: 0
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_ch_bw_mhz: 50
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_rsrq_db: -11
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_sinx_db: 25.5
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_band: n78
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_access_technology: 5G
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_tac: 100
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_tx_channel: 632628
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_cell_id: 500
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_dl_mimo: 0
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_rx_channel: 632628
2022-03-03 12:02:03 - INFO - LOG.Log.LogFile - radio_nr_rssi_dbm: -58

```

## Data Quality Testing

## Radio Quality Testing

Figure 3.9.16: Log file contains results of iPerf and radio tests

The test case will provide end-to-end network performance KPIs related to (log sample is shown below):

- Data Quality Testing (DQT) modules
  - Web
  - DNS
  - Ping
  - HTTP/FTP DL
  - FTP UL
  - Iperf
- Radio Quality Testing (RQT)
  - Full 5G KPI support for NSA/SA operation on x86-based industrial PC with SieraWireless EM9191 5G modem)

### 3.9.5 NetApp requirements for testbeds

NetApp #9 requirements for the implementation and correct operation within the testbed where it will be deployed are the following:

- OSM 10 with modified RO (resource orchestrator) to allow selecting flavors with extra specs (needed for PCI PASSTHROUGH)
- OpenStack flavor containing extra specs (PCI PASSTHROUGH:ALIAS, HW:CPU\_SOCKETS, HW:CPU\_CORES, HW:CPU\_THREADS)

- OpenStack host with SDR card exposed to OSM

### 3.10. NetApp #10

#### 3.10.1 NetApp functional overview, current status and next steps

Neobility's NetApp, Vehicle Route Optimizer (commercially "NeoBus") is a dynamic pooled transportation service (Demand Responsive Transport (DRT)) solution based on minibuses ( $\pm 10$  seats) that is as flexible as traditional ride-sharing, while being a fraction of the price. High level architecture diagram of the NetApp is depicted in the Figure 3.10.1.

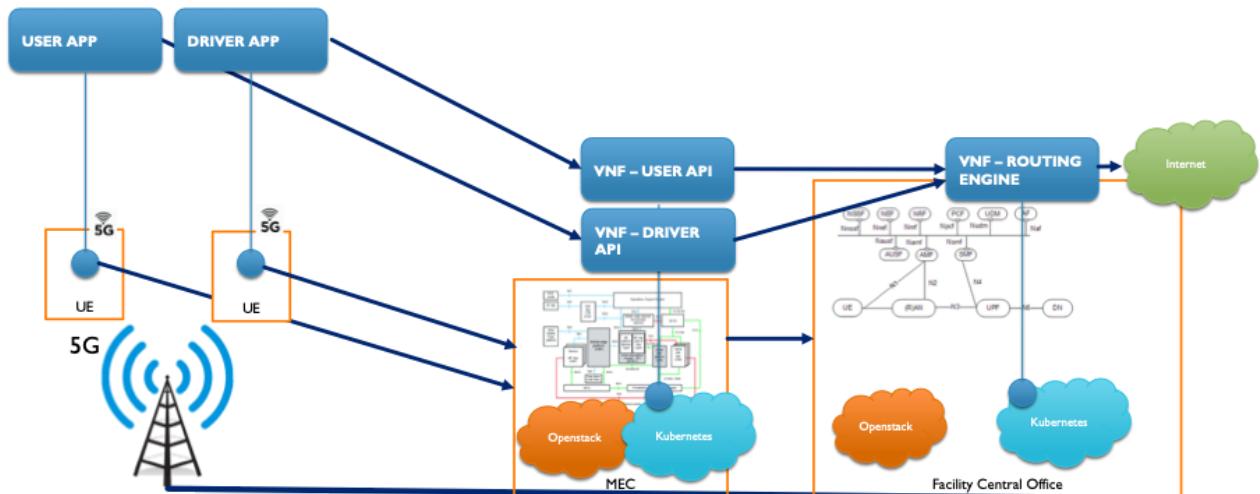


Figure 3.10.1: High level architecture diagram of the NeoBus NetApp

Docker containerization of the services (Dockerfiles) has been prepared and packaged as VNFs:

- User API ("main-api")
- Driver API ("transport-api")
- Routing Engine ("routing-engine")

The API Server and Bus API Server are being written in Typescript with Node.js and are packaged as Docker containers.

The Vehicle Route Optimizer is being written in Java 11 and packaged as a Docker container.

Figure 3.10.2 shows the Docker images creation and run.

```

#!/bin/bash

# OSRM requirement for the "routing-engine"
docker run --detach --env APP_ENV=dev --publish 5000:5000 neobility/osrm-backend-docker:latest

# NeoBus (Transport) API Server
docker run --detach --env NODE_ENV=dev --publish 3001:3000 neobility/transport-api:latest

# Routing engine
docker run --detach --env APP_ENV=dev --publish 3002:3000 neobility/routing-engine:latest

# API Server for the UrbanAir App
docker run --detach --env NODE_ENV=dev --env DEPLOYMENT=dev --publish 3003:3000 neobility/main-api:latest

```

Figure 3.10.2: “Docker run” commands to create the Docker containers

### 3.10.2 NetApp analysis

In this subsection, the NSDs and VNFDs of the NetApp are presented from the point of view of the analysis. Note that services required for the three main components presented are omitted in this section, since they are not a direct part of the NetApp – OSRM service, PostgreSQL database, Redis cache, etc.

#### 3.10.2.1 NSDs

VRO NSD	
Number of Services (NSDs)	1
Number of total VNFs	3
Dependencies of the 5G System	Dynamic VM deployment and MANO connectivity to recover deployment status information
How is it operated	Automated deployment and provisioning
Dependencies	No

Table 3.10.1: VRO NSD

#### 3.10.2.2 VNFs

VNF User API	
Packaging Info:	VNF/CNF (OSM10)
Placement (latency from-to)	Latency from UE: max 20ms
Internet access	Yes
Resource req	2 vCPU, 4GB RAM, 20GB HDD
Delivery model	Container

Table 3.10.2: VNF User API

VNF Driver API	
Packaging Info:	VNF/CNF (OSM10)
Placement (latency from-to)	Latency from UE: max 20ms

Internet access	Yes
Resource req	2 vCPU, 4GB RAM, 20GB HDD
Delivery model	Container

Table 3.10.3: VNF Driver API

VNF Routing Engine	
Packaging Info:	VNF/CNF (OSM10)
Placement (latency from-to)	Latency from UE: max 20ms
Internet access	Yes
Resource req	4 vCPU, 16GB RAM, 50GB HDD
Delivery model	Container

Table 3.10.4: VNF Routing Engine

### 3.10.2.3 NSD and VNF relations/dependencies

The NetApp consists of a single network service that contains the three VNFs. The services communicate between themselves, the User API and the Driver API are dependent of the Routing Engine.

### 3.10.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - Containers (User API, Driver API and Routing Engine).
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs:
  - Every container will expose a healthcheck endpoint. The deployment is successful when all containers are healthy.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - Requires dynamic VM deployment and MANO connectivity to recover deployment status information.
- How it is operated (and if it requires manual interventions):
  - Automated deployment and provisioning.
- Dependencies (does it expose or consume services from/to other NetApps): No.

VRO NetApp NEST	
Area of service	RO
Area of service: Region specification	Bucharest
Downlink maximum throughput per UE	-
Uplink maximum throughput per UE	-
Isolation level	Virtual resources isolation
Slice quality of service parameters: 3GPP 5QI	70

Maximum Packet Loss Rate	1%
Supported Device Velocity	3: Vehicular: 10 km/h to 120 km/h

Table 3.10.5: VRO NetApp's NEST

### 3.10.3.1 NSDs definitions

```

nsd:
  nsd:
    - id: VRO_OSM10_nsd
      name: VRO_OSM10_nsd
      description: VRO OSM10 NSD
      designer: NEO
      version: '1.0'
      df:
        - id: default-df
      vnf-profile:
        - id: '1'
          vnfd-id: user_api_vnfd
          virtual-link-connectivity:
            - constituent-cpd-id:
              - constituent-base-element-id: 'user_api'
                constituent-cpd-id: mgmt-vnf-ext
                virtual-link-profile-id: 5GASP
            - id: '2'
              vnfd-id: driver_api_vnfd
              virtual-link-connectivity:
                - constituent-cpd-id:
                  - constituent-base-element-id: 'driver_api'
                    constituent-cpd-id: mgmt-vnf-ext
                    virtual-link-profile-id: 5GASP
            - id: '3'
              vnfd-id: routing_engine_vnfd
              virtual-link-connectivity:
                - constituent-cpd-id:
                  - constituent-base-element-id: '1'
                    constituent-cpd-id: mgmt-vnf-ext
                    virtual-link-profile-id: 5GASP
              virtual-link-desc:
                - id: 5GASP
                  mgmt-network: 'true'
              vnfd-id:
                - user_api_vnfd
                - driver_api_vnfd
                - routing_engine_vnfd

```

### 3.10.3.2 VNFs definitions

```

vnfd:
  description: User API
  id: user_api_vnfd
  product-name: user_api_vnfd
  provider: NEO
  version: '1.0'
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-vnf-ext
      k8s-cluster-net: mgmt-net
      mgmt-cp: mgmt-vnf-ext
      k8s-cluster:
        nets:
          - id: mgmt-vnf-ext
      kdu:

```

```

- name: user_api
  helm-chart: neobility/main-api

```

```

vnfd:
  description: Driver API
  id: driver_api_vnfd
  product-name: driver_api_vnfd
  provider: NEO
  version: '1.0'
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-vnf-ext
      k8s-cluster-net: mgmt-net
    mgmt-cp: mgmt-vnf-ext
    k8s-cluster:
      nets:
        - id: mgmt-vnf-ext
  kdu:
    - name: driver_api
      helm-chart: neobility/transport-api

```

```

vnfd:
  description: Routing Engine API
  id: routing_engine_vnfd
  product-name: routing_engine_vnfd
  provider: NEO
  version: '1.0'
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-vnf-ext
      k8s-cluster-net: mgmt-net
    mgmt-cp: mgmt-vnf-ext
    k8s-cluster:
      nets:
        - id: mgmt-vnf-ext
  kdu:
    - name: routing_engine
      helm-chart: neobility/routing-engine

```

### 3.10.4 NetApp's tests plan and definition

The NetApp will follow the test plan defined in WP5 [3]. WP5 defines the test cases to be performed. For the VRO NetApp, the selected mandatory tests are 4, 19 and the optional tests 2, 8, 13, 14, 15 and 18. The NetApp will also do some performance tests: latency, packet loss and jitter.

The functional testing of the NetApp will consist of deploying the services and simulating route generations.

A successful deployment will be validated by checking the health endpoints of every component.

Simulating route generations will use a list of mocked locations of users and drivers. KPIs related to the route generation will be collected and compared to some thresholds: distance of the route, the order of pick-ups and drop-offs, etc. If the values are above or below the thresholds will determine if the functional test has passed or not.

### 3.10.5 NetApp requirements for testbeds

The NetApp requires to have connectivity to UEs consisting of multiple (two or more) 5G Android smartphones, since the “User App” and “Driver App” are Android applications. Also, the testbed needs to have coverage outside of the facility.

This is required to make a real “physical” test, where the apps are installed on the smartphones, connected to the testbed via 5G connectivity, and multiple persons (two or more) will need to go outside of the facility and follow a testing procedure.

## 3.11. NetApp #11

### 3.11.1 NetApp functional overview, current status and next steps

An objective of the FIDEGAD NetApp is to provide a cloud native application onboarded to an air drone which can allow teams on ground to see the drone’s live images and detect fires. The system being developed consists of a client on the drone which acquires the video input and forwards it to the image recognition server which detects the existence of fire. Currently, the image recognition server resides at the 5G Core (Phase I), but it is expected that it will be hosted at the edge (Phase II), so as to provide significantly lower latency times. Also, the control center service is currently packaged with the image recognition server, allowing access to the video stream acquired from the drone via a web server. Eventually, the control center would be seen as a standalone service, further enhancing the acquisition of control of the ground units.

The architectural diagrams of the NetApp can be found in Figure 3.11.1 and Figure 3.11.2.

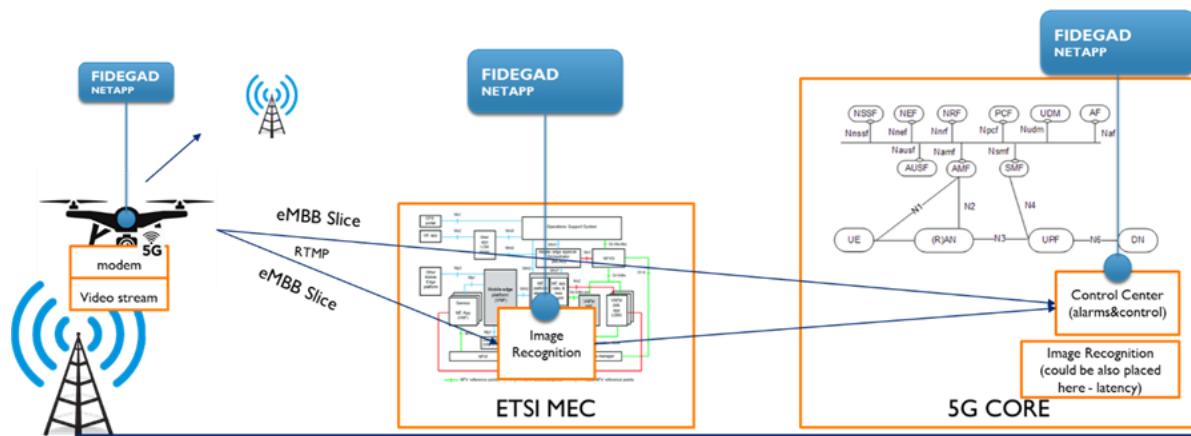


Figure 3.11.1: High level architecture diagram of the FIDEGAD NetApp

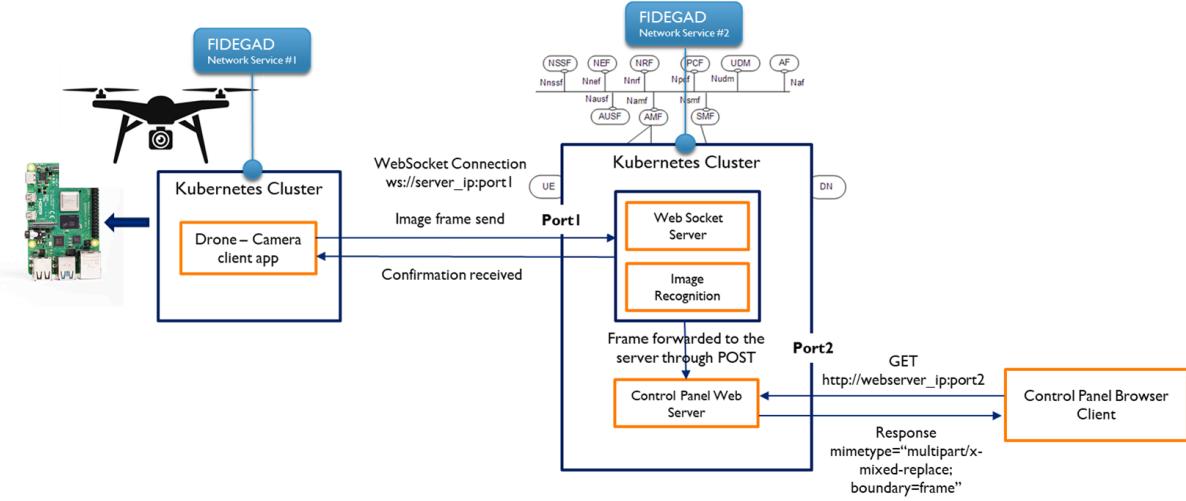


Figure 3.11.2: Architectural diagram of the FIDEGAD NetApp

Helm charts and Docker images for the NetApp components have been prepared and packaged as NS/CNF. These components are:

- cameraastr - Backend AI fire detection and streaming server
- cameraastr\_client - Client application installed on drone

The respective Docker images have been uploaded to a private Docker registry. Likewise, Helm Charts have been uploaded to a private Helm Chart repository. The consequent NSDs/VNFDs has been successfully onboarded to OSM10.

Currently, there are two configuration parameters exposed for the instantiating (Day-1) configuration regarding the cameraastr\_client, namely "input1" and "output1". Specifically:

- input1: Defines the input provided to the cameraastr\_client. This can be an RTSP stream from an IP camera or an image input device available on the client (for example /dev/video0)
- output1: The URL of the cameraastr backend AI fire detection and streaming server, where the images from the cameraastr\_client will be sent to.

Following, there is an OSM10 instantiation example for the cameraastr\_client NS with Day-1 configuration:

```
osm ns-create --ns_name cameraastr_client_ns_test --nsd_name cameraastr_client_ns --vim_account CloudVille --config
'{"additionalParamsForVnf":[{"member-vnf-index": "1",
"additionalParamsForKdu":[{"kdu_name":"cameraastr_client","additionalParams": {"input1":
"rtsp://127.0.0.1/local/stream.mp4", "output1":"ws:// 10.10.10.100:8765"}]}]}]
```

Concerning the cameraastr NS, there are not any configurable parameters for Day-1 operations.

Finally, several initial development tests have been performed for the current implementation of the system, and are presented below:

- using existing video streams of fires as input

- using a Raspberry Pi 3 with a USB camera, emulating the IP camera providing an input video stream to the cameraastr\_client ns
- using a PC with a USB camera as a cloud native application and privileged container configuration (the PC working as a VIM registered to the OSM10)

In relation to the NetApp design, implementation of the first version of the VNFD and NSD descriptors, and development of VNF/NS packaging and onboarding to OSM 10 at Patras testbed has been done (see Figure 3.11.3, Figure 3.11.4, Figure 3.11.5, Figure 3.11.6). Additionally, employing the CI/CD pipeline as described in NetApp's tests plan below (see Section 3.11.4) and eventually further development to align with Phase II requirements will follow.

```
ubuntu@osm-ten-2g:~$ osm ns-create --ns_name cameraastr_client_test3 --nsd_name cameraastr_client_ns --vim account CloudVille --config '{"additionalParamsForVnf":[{"member-vnf-index":"1","additionalParamsForKdu":[{"kdu_name":"cameraastr-client","additionalParams":{"input1":"rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mp4","output1":"ws://10.10.10.170:8765"}]}]}' 2d67da74-ab93-470c-9196-85a50e6e528a
```

Figure 3.11.3: NS instantiation command with Day-1 parameters

NS Instances						
Init	running / configured	failed	scaling	Actions		
cameraastr_client_ns	2d67da74-ab93-470c-9196-85a50e6e528a	cameraastr_client_ns	✓	✓	Done	Action
cameraastr_ns	02be94b-6109-4ace-84ef-258c52b765a	cameraastr_ns	✓	✓	Done	Action

Figure 3.11.4: Running NS instantiated with Day-1 parameters

```
ubuntu@osm10-k8s-master:~$ kubectl get services --all-namespaces
NAMESPACE          NAME           TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)         AGE
84e1641f-3a6e-469e-ac50-be979ce99c55 nam-cameraastr-0-11-0-0032557648 LoadBalancer 10.97.181.20  10.10.10.170  85:30594/TCP,8765:30667/TCP 6m4s
84e1641f-3a6e-469e-ac50-be979ce99c55 nam5-cameraastr-client-0-1-0-0035759794 ClusterIP   <none>        80/TCP          2m26s
```

Figure 3.11.5: Running Kubernetes services

```
ubuntu@osm10-k8s-master:~$ kubectl get deployment --all-namespaces
NAMESPACE          NAME           READY   UP-TO-DATE   AVAILABLE   AGE
84e1641f-3a6e-469e-ac50-be979ce99c55 nam-cameraastr-0-11-0-0032557648 1/1     1           1           7m19s
84e1641f-3a6e-469e-ac50-be979ce99c55 nam5-cameraastr-client-0-1-0-0035759794 1/1     1           1           3m41s
```

Figure 3.11.6: Running Kubernetes deployments

### 3.11.2 NetApp analysis

In this subsection, the NSDs and VNFDs of FIDEGAD NetApp are presented from the point of view of the analysis. These NFV artifacts cover the current Phase I version of the NetApp.

#### 3.11.2.1 NSDs

FIDEGAD NetApp comprises of two Network Services, described by their respective Network Service Descriptors (NSDs). Network services' functionality are presented in the previous section.

Cameraastr NSD	
Number of Services (NSDs)	1
Number of total VNFs	1

Dependencies of the 5G System	1 network slice (eMBB type)
How is it operated	Automatically deployed and provisioned via Helm Charts
Dependencies	None

Table 3.11.1: Cameraaistr NSD

Cameraaistr_client NSD	
Number of Services (NSDs)	1
Number of total VNFs	1
Dependencies of the 5G System	1 network slice (eMBB type)
How is it operated	Automatically deployed and provisioned via Helm Charts
Dependencies	None

Table 3.11.2: Cameraaistr client NSD

### 3.11.2.2 VNFs

The implementation of described Network Services is achieved by VNFs which are defined as follows. In that respect, two VNFs have been created for the FIDEGAD NetApp to implement the respective network services defined in the previous section.

Cameraaistr VNF	
Component	Cameraaistr NS
Packaging Info:	VNF/CNF (OSM10)
Placement (latency from-to)	<100ms (placed at 5G Core)
Internet access	Yes
Minimum resource requirements	4 Cores, 16GB Memory, No storage
Delivery model	Container
Orchestration	VNF/Helm Chart

Table 3.11.3: Cameraaistr VNF

Cameraaistr_client VNF	
Component	Cameraaistr_client NS
Packaging Info:	VNF/CNF (OSM10)
Placement (latency from-to)	<10ms (placed at 5G Core)
Internet access	No
Minimum resource requirements	2 Cores, 1GB Memory, No storage
Delivery model	Container
Orchestration	VNF/Helm Chart

Table 3.11.4: Cameraaistr client VNF

### 3.11.2.3 NSD and VNF relations/dependencies

The relations and dependencies among the different VNFs, implementing the respective network services of the NetApp are depicted in Figure 3.11.2. FIDEGAD comprises of two network services, Cameraastr\_client NS charged with the acquisition and the streaming of video input from the UE and Cameraastr NS, which performs the image recognition and exposes the video stream through a web server.

### 3.11.3 NetApp definition

- Packaging Info (Virtual Machine (VM), container, type, etc.):
  - NSD/VNFD templates (OSM10)
  - 2 NSDs: Cameraastr, Cameraastr\_client
- Existing Healthcheck or Lifecycle (deployment status) Hooks or APIs: No.
- Dependencies of the 5G System (requires slicing, core functionality, etc.):
  - An eMBB network slice
- How is it operated (and does it require manual interventions): Automated deployment and provisioning.
- Dependencies (does it expose or consume services from/to other NetApps): No.

FIDEGAD NetApp NEST	
Area of service	GR
Area of service: Region specification	Patras
Downlink maximum throughput per UE	1 Mbps
Uplink maximum throughput per UE	5 Mbps
Isolation level	Virtual resources isolation
Slice quality of service parameters: 3GPP 5QI	69
Supported Device Velocity	0-10 km/h
User data access	Termination in the private network

Table 3.11.5: FIDEGAD NetApp's NEST

#### 3.11.3.1 NSDs definitions

##### Cameraastr\_client NSD:

```
nsd:  
  nsd:  
    - description: NS for cameraastr_client KNF connected to mgmt network  
    designer: UoP  
    df:  
      - id: default-df  
      vnf-profile:  
        - id: '1'  
        virtual-link-connectivity:  
          - constituent-cpd-id:  
            - constituent-base-element-id: cameraastr_client  
            constituent-cpd-id: mgmt-ext  
            virtual-link-profile-id: adminnet1
```

```

vnfd-id: cameraastr_client_knf
id: cameraastr_client_ns
name: cameraastr_client_ns
version: '2.0'
virtual-link-desc:
  - id: adminnet1
    mgmt-network: 'true'
vnfd-id:
  - cameraastr_client_knf

```

#### Cameraastr NSD:

```

nsd:
  nsd:
    - description: NS for cameraastr KNF connected to mgmt network
      designer: UoP
    df:
      - id: default-df
    vnf-profile:
      - id: '1'
        virtual-link-connectivity:
          - constituent-cpd-id:
            - constituent-base-element-id: cameraastr
              constituent-cpd-id: mgmt-ext
              virtual-link-profile-id: adminnet1
        vnfd-id: cameraastr_knf
    id: cameraastr_ns
    name: cameraastr_ns
    version: '2.0'
    virtual-link-desc:
      - id: adminnet1
        mgmt-network: 'true'
    vnfd-id:
      - cameraastr_knf

```

#### 3.11.3.2 VNFs definitions

##### Cameraastr\_VNFD:

```

vnfd:
  description: KNF with single KDU with cameraastr using a helm-chart
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-ext
      k8s-cluster-net: mgmtnet
  id: cameraastr_knf
  k8s-cluster:
  nets:
    - id: mgmtnet
  kdu:
    - name: cameraastr
      helm-chart: NAM/cameraastr:0.11.0
  mgmt-cp: mgmt-ext
  product-name: cameraastr_knf
  provider: UoP
  version: '2.0'

```

##### Cameraastr\_client\_VNFD:

```

vnfd:
  description: KNF with single KDU with cameraastr_client using a helm-chart
  df:
    - id: default-df

```

```

ext-cpd:
  - id: mgmt-ext
    k8s-cluster-net: mgmtnet
    id: cameraastr_client_knf
    k8s-cluster:
      nets:
        - id: mgmtnet
    kdu:
      - name: cameraastr_client
        helm-chart: NAM/cameraastr-client:0.1.0
    mgmt-cp: mgmt-ext
    product-name: cameraastr_client_knf
    provider: UoP
    version: '2.0'

```

### 3.11.4 NetApp's tests plan and definition

The NetApp will follow the common test plan defined in WP5/D5.3 [3] in particular, with the selected test cases to be performed, as presented below:

- mandatory tests: 4, 19
- optional tests: 2, 7, 13

Since NSD/VNFD artifacts are created following the NFV model (SOL006), and successfully passed the static validation process during the onboarding procedure, the deployment in an environment simulating the nominal use case is targeted. Regarding the functional testing of the NetApp in that environment, a deployment employing both software and hardware parts of the testbed will be attempted, so as to validate the NetApp's automation goals for seamless orchestration.

Such a scenario would be fulfilled by deploying the image recognition server in a hosting 5G network slice and consequently the video stream client in a respective device. The expected outcome is the provision of the client's application, which will provide the video stream, and the forwarding of the latter towards the end user through the monitoring application, after a fire detection incident.

The test case will be considered successful, should the communication channel remain stable and provide adequate level of fire detection.

### 3.11.5 NetApp requirements for testbeds

FIDEGAD NetApp's core requirement for the testbed is the provisioning of a 5G network slice. In relation to software, an installation of OSM 10 is required, as the NetApp's artifacts are designed towards this specific version. Concerning the virtualized infrastructure management, an instance of Kubernetes must be present to facilitate the deployment of NetApp's network service related to the testbed. Regarding the hardware requirements, a universal client hosting a Kubernetes installation, also bearing a camera device, would suffice. In the nominal use case scenario, a Raspberry Pi 4, being adequate to support a Kubernetes installation with a camera module, is used. The device, facilitating the deployment of client-side network service, is then fitted onto the drone.

## 4. Conclusions

The deliverable presents the current status and progress of the NetApps' design and development. The process of the NetApp creation has been defined by the methodology presented in deliverables D4.1 [1] and D4.2 [2]. Therefore, to report on the current status of each single NetApp, this deliverable's structure follows guidelines introduced by the methodology.

Besides giving high-level description of every NetApp's status and next steps planned, the deliverable aims to go into certain technical details by presenting NSDs and VNFs definitions, NEST required by the NetApp, NetApp's deployment plan, test plan and requirements for testbeds. Where possible, descriptions and definitions are supported by extracts from configuration files and screenshots representing certain functions execution on the system.

As one can notice throughout the deliverable, NetApps have reached different states of the progress. There are various reasons, however, one of the most obvious is the design and development is faced with certain challenges and/or issues. Such cases are documented and any issues elaborated. It might be worth mentioning the challenges are discussed within the consortium members seeking for optimal solutions and, also, to be aligned with the deadlines set.

As already mentioned in the Introduction section, this deliverable reports on initial/midterm status of NetApps, while final report on NetApp design development will follow in *D4.4 Development of NetApps in the Automotive and PPDR Verticals*.

## Bibliography

- [1] 5GASP, "D4.1 Initial Methodology for Designing and Developing NetApps," 2021.
- [2] 5GASP, "D4.2 Final Methodology for Designing and Developing NetApps," 2021.
- [3] 5GASP, "D5.3 Final test specification on test-plan creation and testing methodologies," deliverable in preparation, 2022.
- [4] "TMF 633 – Service Catalog Management," TM Forum.
- [5] 5GASP, "D3.1 5GASP experimentation services, middleware and multi-domain facilities continuous integration," 2021.
- [6] M. Dawe, "Although containers have existed in various formats for the last thirty years, the popularity of containers has grown exponentially since Docker emerged in 2013," 2021. [Online]. Available: <https://www.capgemini.com/se-en/2021/09/containerization-why-now/>. [Accessed 2022].
- [7] "Red Hat Ansible Automation Platform," [Online]. Available: <https://www.ansible.com/>. [Accessed 2022].
- [8] 5GASP, "D2.1 The 5GASP reference architecture," 2021.
- [9] IEEE, "Standard for Information Technology -- Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks -- Specific Requirements - Part 11: Wireless Local Area Network (LAN) Medium Access Control (MAC) and Physical Layer (P)," 2010.
- [10] ETSI, "N 302 663 - Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band," 2020.
- [11] 5GASP, "D5.2 Integration guide and API reference manual," 2022.