Programmable edge-to-cloud virtualization fabric for the 5G Media industry

D2.3: 5G-MEDIA Platform Architecture

Work Package: WP2 – Architecture, Analysis and Tools

Lead partner: SiLO

Francesco Iadanza [ENG], Pasquale Andriani [ENG], David Breitgand [IBM], Avi Weit [IBM], Stamatia Rizou [SiLO], Panagiotis Athanasoulis [SiLO], George Agapiou [OTE], Christina Lessi [OTE], Christos Tsirakis [OTE], Óscar Prieto [RTVE], Gabriel Solsona [RTVE], David Griffin [UCL], Khoa Phan [UCL], Morteza Kheirkhah [UCL], Miguel Rio [UCL], David Lopez Meco [TID], Rocio Ortiz Pena [TID], Alberto Florez Pages [TID], David Jiménez [UPM], Federico Alvarez [UPM], Javier Serrano [UPM], José Manuel Menéndez [UPM], Gordana Polanec-Kutija [IRT], Madeleine Keltsch [IRT], Gino Carrozzo [NXW], Francesca Moscatelli [NXW], Elian Kraja [NXW], Nikolaos Zioulis [CERTH], Dimitris Zarpalas [CERTH], Ugur Acar [NET], Cengiz Gezer [NET], Igor Fritzsch [BIT]

Author(s):

Delivery date (DoA): August 31st, 2018
Actual delivery date: August 2nd, 2018
Dissemination level: Public
Version number: 1.0 Status: Final

Grant Agreement N°: 761699
Project Acronym: 5G-MEDIA
Project Title: Programmable edge-to-cloud virtualization fabric for the 5G Media industry
Instrument: IA
Call identifier: H2020-ICT-2016-2
Topic: ICT-08-2017, 5G PPP Convergent Technologies, Strand 2: Flexible network applications
Start date of the project: June 1st, 2017
Duration: 30 months

Project co-funded by the European Commission under the Horizon 2020 Programme.
### Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Who</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Dec. 8th 2017</td>
<td>SiLO</td>
<td>ToC</td>
</tr>
<tr>
<td>0.15</td>
<td>Jan. 8th 2018</td>
<td>SiLO, NET, ENG, UPM, IRT, BIT, NXW, OTE, UCL, TID, IBM, RTVE, CERTH</td>
<td>Relative technologies and 5G-PPP phase 1 outcomes. First draft of 5G-MEDIA platform software architecture. First draft of 5G-MEDIA NSs and VNFs</td>
</tr>
<tr>
<td>0.2</td>
<td>Jan 24th 2018</td>
<td>SiLO, NET, NXW, UCL, ENG</td>
<td>Update 5G-MEDIA platform software architecture</td>
</tr>
<tr>
<td>0.3</td>
<td>May 14th 2018</td>
<td>SiLO</td>
<td>Refined structure</td>
</tr>
<tr>
<td>0.4</td>
<td>June 7th 2018</td>
<td>IRT, ENG, SiLO, NXW, OTE, UCL, IBM, NETAS</td>
<td>Update list of 5G-MEDIA VNFs/PNFs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update 5G-MEDIA ASDK &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update monitoring service &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update 5G-MEDIA Catalogue &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update exploitation of C-RAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update CNO &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update VIM for OW &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update 5G-MEDIA ASDK &amp; seq. diagrams</td>
</tr>
<tr>
<td>0.5</td>
<td>July 5th 2018</td>
<td>SiLO</td>
<td>Address comments from internal reviews</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update executive summary, introduction, conclusions</td>
</tr>
<tr>
<td>0.6</td>
<td>July 25th 2018</td>
<td>IBM, TID</td>
<td>Update serverless computing &amp; seq. diagrams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update specs of OpenNebula integration</td>
</tr>
<tr>
<td>0.7</td>
<td>July 26th 2018</td>
<td>IBM, ENG, CERTH, NETAS</td>
<td>Minor changes</td>
</tr>
<tr>
<td>1.0</td>
<td>July 27th 2018</td>
<td>SiLO, ENG</td>
<td>Finalize document for submission</td>
</tr>
</tbody>
</table>
## Quality Control

<table>
<thead>
<tr>
<th>Role</th>
<th>Date</th>
<th>Who</th>
<th>Approved/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Review</td>
<td>June 18th 2018</td>
<td>NXW</td>
<td>Approved with minor comments addressed in the final version</td>
</tr>
<tr>
<td>Internal Review</td>
<td>June 26th 2018</td>
<td>ENG</td>
<td>Approved with minor comments addressed in the final version</td>
</tr>
</tbody>
</table>
Disclaimer

This document may contain material that is copyright of certain 5G-MEDIA project beneficiaries and may not be reproduced or copied without permission. The commercial use of any information contained in this document may require a license from the proprietor of that information. The 5G-MEDIA project is part of the European Community's Horizon 2020 Program for research and development and is as such funded by the European Commission. 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 user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability with respect to this document, which is merely representing the authors' view.

The 5G-MEDIA Consortium is the following:

<table>
<thead>
<tr>
<th>Participant number</th>
<th>Participant organisation name</th>
<th>Short name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>ENGINEERING – INGEGNERIA INFORMATICA SPA</td>
<td>ENG</td>
<td>Italy</td>
</tr>
<tr>
<td>02</td>
<td>IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD</td>
<td>IBM</td>
<td>Israel</td>
</tr>
<tr>
<td>03</td>
<td>SINGULARLOGIC ANONYMI ETAIREIA PLIROFORIAKON SYSTIMATON KAI EFARMOGON PLIROFORIKIS</td>
<td>SILO</td>
<td>Greece</td>
</tr>
<tr>
<td>04</td>
<td>HELLENIC TELECOMMUNICATIONS ORGANIZATION S.A. - OTE AE (ORGANISMOS TILEPIKIONION TIS ELLADOS OTE AE)</td>
<td>OTE</td>
<td>Greece</td>
</tr>
<tr>
<td>05</td>
<td>CORPORACION DE RADIO Y TELEVISION ESPANOLA SA</td>
<td>RTVE</td>
<td>Spain</td>
</tr>
<tr>
<td>06</td>
<td>UNIVERSITY COLLEGE LONDON</td>
<td>UCL</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>07</td>
<td>TELEFONICA INVESTIGACION Y DESARROLLO SA</td>
<td>TID</td>
<td>Spain</td>
</tr>
<tr>
<td>08</td>
<td>UNIVERSIDAD POLITECNICA DE MADRID</td>
<td>UPM</td>
<td>Spain</td>
</tr>
<tr>
<td>09</td>
<td>INSTITUT FUER RUNDFUNKTECHNIK GMBH</td>
<td>IRT</td>
<td>Germany</td>
</tr>
<tr>
<td>10</td>
<td>NEXTWORKS</td>
<td>NXW</td>
<td>Italy</td>
</tr>
<tr>
<td>11</td>
<td>ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS</td>
<td>CERTH</td>
<td>Greece</td>
</tr>
<tr>
<td>12</td>
<td>NETAS TELEKOMUNIKASYON ANONIM SIRKETI</td>
<td>NET</td>
<td>Turkey</td>
</tr>
<tr>
<td>13</td>
<td>INTERINNOV SAS</td>
<td>IINV</td>
<td>France</td>
</tr>
<tr>
<td>14</td>
<td>BITTUBES GMBH</td>
<td>BIT</td>
<td>Germany</td>
</tr>
<tr>
<td>15</td>
<td>NATIONAL CENTER FOR SCIENTIFIC RESEARCH &quot;DEMOKRITOS&quot;</td>
<td>NCSRD</td>
<td>Greece</td>
</tr>
</tbody>
</table>
Table of Contents

EXECUTIVE SUMMARY ........................................................................................................... 14
1. INTRODUCTION .................................................................................................................. 16
  1.1. Scope of this deliverable ................................................................................................. 16
2. CORE TECHNOLOGY AND STATE-OF-THE-ART .......................................................... 18
  2.1. Projects and offerings of 5G-PPP Phase 1 ...................................................................... 18
    2.1.1. The project COGNET ............................................................................................. 18
    2.1.2. The project SELFNET .......................................................................................... 24
    2.1.3. The project SONATA ............................................................................................ 28
  2.2. Tools and solutions transferred to 5G-MEDIA by partners ............................................ 32
    2.2.1. OnLife platform .................................................................................................... 32
    2.2.2. C-RAN .................................................................................................................. 37
    2.2.3. Elastic transcoder and interactive 360° media platform ......................................... 39
    2.2.4. Serverless computing ............................................................................................ 41
    2.2.5. Symphony ............................................................................................................. 47
  2.3. Summary of exploitable tools and solutions ................................................................. 49
3. EVALUATION OF CANDIDATE OPEN SOURCE SOLUTIONS FOR 5G-MEDIA PLATFORM 52
  3.1. Evaluation of service development kit tools ................................................................. 52
    3.1.1. SDK candidate solutions ....................................................................................... 52
    3.1.2. Evaluation criteria, assessment and reasoning ....................................................... 60
  3.2. Evaluation of MANO frameworks .................................................................................. 65
    3.2.1. MANO Candidate solutions ............................................................................... 65
    3.2.2. Evaluation criteria, assessment and reasoning ........................................................ 81
4. 5G-MEDIA PLATFORM SOFTWARE ARCHITECTURE ..................................................... 94
  4.1. High level architecture of 5G-MEDIA platform ............................................................ 94
  4.2. Service (Application) development kit .......................................................................... 96
    4.2.1. Cross reference of 5G-MEDIA SDK to SONATA SDK ........................................... 99
    4.2.2. DevOps continuous integration and deployment services for the SDK .................... 100
  4.3. Service virtualization platform and core components .................................................. 101
    4.3.1. 5G-MEDIA MANO framework ............................................................................. 101
    4.3.2. The 5G-MEDIA catalogue ................................................................................... 103
    4.3.3. Media service MAPE ............................................................................................ 106
  4.4. Extensions in virtualized infrastructure managers ....................................................... 111
    4.4.1. Virtualized infrastructure manager for OnLife ....................................................... 111
4.4.2. Virtualized infrastructure manager for OpenWhisk .............................................................. 114

4.5. SEQUENCE DIAGRAMS FOR SELECTED OPERATIONS ......................................................... 117

4.5.1. VNF development cycle ...................................................................................................... 117

4.5.2. NS Emulation in SDK .......................................................................................................... 118

4.5.3. VNF package onboarding in the Catalogue .......................................................................... 119

4.5.4. FaaS development cycle ..................................................................................................... 121

4.5.5. NS onboarding in the Catalogue .......................................................................................... 123

4.5.6. NS Instantiation ................................................................................................................... 124

4.5.7. NS scale out running FaaS .................................................................................................. 125

4.5.8. VNF forwarding graph optimization in runtime ................................................................. 126

4.5.9. NS monitoring in MAPE loop .............................................................................................. 128

4.5.10. Catalogue authentication and authorization ........................................................................ 129

4.5.11. Accounting ......................................................................................................................... 129

5. 5G-MEDIA FUNCTIONS AND SERVICES PORTFOLIO ......................................................... 131

5.1. ETSI VNF PACKAGE SPECIFICATIONS ................................................................................ 131

5.2. LIST OF VNFS IN 5G-MEDIA PROJECT ............................................................................... 132

6. CONCLUSIONS .......................................................................................................................... 136

7. ANNEX I – POTENTIALLY EXPLOITABLE RESOURCES OF 5G PPP PHASE 1 PROJECTS... 137

7.1. THE PROJECT COHERENT .................................................................................................... 137

7.1.1. Overview ............................................................................................................................. 137

7.1.2. Technical approach ............................................................................................................ 138

7.1.3. Innovation ........................................................................................................................ 139

7.2. THE PROJECT CHARISMA .................................................................................................. 139

7.2.1. Overview ............................................................................................................................. 139

7.2.2. Technical approach ............................................................................................................ 140

7.2.3. Innovation ........................................................................................................................ 142

7.3. THE PROJECT SESAME....................................................................................................... 143

7.3.1. Overview ............................................................................................................................. 143

7.3.2. Technical approach ............................................................................................................ 144

7.3.3. Innovation ........................................................................................................................ 145

8. ANNEX II – LIST OF 5G-MEDIA VIRTUAL NETWORK FUNCTIONS ............................... 147

8.1. APPLICATION LAYER FUNCTIONS ....................................................................................... 147

8.2. GENERIC MEDIA FUNCTIONS ............................................................................................. 149

8.3. NETWORK LAYER FUNCTIONS ............................................................................................ 149
### List of Figures

- Figure 1 – CogNet solution .......................................................... 19
- Figure 2 – Logical components of CogNet platform .......................... 21
- Figure 3 – SELFNET system architecture ........................................ 25
- Figure 4 – Network function types .................................................. 26
- Figure 5 – The architecture of SONATA SP ....................................... 30
- Figure 6 – Logical architecture and components of the CTpd project .......... 34
- Figure 7 – Service architecture of the CTpd project ............................ 36
- Figure 8 – (left) Legacy RAN architecture, (right) C-RAN architecture ........ 38
- Figure 9 – Pareto Front of different converged architectures .................. 38
- Figure 10 – OTEs IP RAN architecture ............................................ 39
- Figure 11 – BitTubes elastic transcoder ........................................... 40
- Figure 12 – Interactive 360° media platform ..................................... 41
- Figure 13: OpenWhisk (and FaaS) in a nutshell ................................ 42
- Figure 14 – OpenWhisk architecture ............................................... 43
- Figure 15 – Lean OpenWhisk Architecture ....................................... 46
- Figure 16 – The Symphony unified software platform .......................... 47
- Figure 17 – Symphony legacy ....................................................... 48
- Figure 18 – Remote building management enabled for authorized terminals .... 48
- Figure 19 – Symphony media component functionalities ....................... 49
- Figure 20 – Validation selection UI .................................................. 55
- Figure 21 – Forwarding graph issue highlight ..................................... 56
- Figure 22 – Forwarding graph cycle identification .............................. 56
- Figure 23 – OSM CI/CD Pipeline ................................................... 59
- Figure 24 – Overview of Open-Baton .............................................. 66
- Figure 25 – Open-Baton functional architecture .................................. 67
- Figure 26 – Mapping between OSM and ETSI NFV MANO ................. 69
- Figure 27 – OPNFV business stack ................................................ 72
- Figure 28 – Upstream projects in OPNFV ......................................... 72
- Figure 29 – Test projects in OPNFV ................................................ 73
- Figure 30 – Mapping of ETSI MANO in OPNFV ................................. 75
Figure 31 – Architecture of SONATA SVP .................................................................78
Figure 32 – MANO framework of SONATA .............................................................79
Figure 33 – Overview of 5G-MEDIA architecture ..................................................94
Figure 34 – Overview of 5G-MEDIA SDK ..............................................................97
Figure 35 – Mapping of son-emu to ETSI NFV MANO ..........................................98
Figure 36 – SDK and main CI/CD services for DevOps ...............................................101
Figure 37 – 5G App and Service Catalogue high level design ..................................105
Figure 38 – The architecture of data acquisition and sharing in Media Service MAPE ...107
Figure 39 – Flow of architecture work .....................................................................109
Figure 40 – Policy/optimization component .............................................................110
Figure 41 – Interaction between policy/optimization component and NFV MANO ...111
Figure 42 – OpenNebula GUI ................................................................................112
Figure 43 – Plugin of OpenNebula in RO of OSM ....................................................112
Figure 44 - Opennebula system interfaces ..............................................................113
Figure 45 – FaaS Integration with MANO ...............................................................115
Figure 46 – Reference software architecture for FaaS VIM ......................................116
Figure 47 – FaaS/MANO sample software architecture ............................................116
Figure 48 – Image building of an application ............................................................118
Figure 49 – VNF packaging and testing .................................................................118
Figure 50 – NS emulation ......................................................................................119
Figure 51 – VNF package onboarding into the Catalogue .........................................121
Figure 52 – On-board VNF FaaS package flow .......................................................122
Figure 53 – Pre-onboarding for FaaS .....................................................................123
Figure 54 – NS onboarding into the catalogue ........................................................124
Figure 55 – Network Service instantiation ..............................................................125
Figure 56 – NS scaling out message flow under FaaS ..............................................126
Figure 57 – Working flow without cognitive network optimizer ..................................127
Figure 58 – Working flow with cognitive network optimizer .....................................128
Figure 59 – Monitoring running NS ....................................................................128
Figure 60 – Catalogue authentication/authorization ................................................129
Figure 61 – Accounting service ............................................................................130
Figure 62 – An example of manifest file ...............................................................132
Figure 63 – Illustration of COHERENT concept ................................................................. 138
Figure 64 – Abstracted network graph ........................................................................... 139
Figure 65 – CHARISMA virtualised C&M converged 5G wireless/wireline infrastructure scenario ................................................................................................................. 141
Figure 66 – SESAME impact on service deployment ....................................................... 143
Figure 67 – SESAME architecture .................................................................................. 144
List of Tables

Table 1 – Exploitation and integration plan of 5G-PPP phase 1 outcomes in 5G-MEDIA........50
Table 2 – Exploitation and integration plan of commercial solutions in 5G-MEDIA ...............50
Table 3 – Assessment of SONATA, Open-Baton and OSM SDKs.................................................62
Table 4 – Assessment of candidate solutions for 5G-MEDIA MANO..........................82
Table 5 – Services and innovation of 5G-MEDIA SDK in contrast to SONATA SDK and OSM Developer Tools.....................................................................................................................100
Table 6 – Main services/actions in the 5G-MEDIA ecosystem ............................................117
Table 7 – List of VNFs implemented in the three UCs of the 5G-MEDIA project .............133
Table 8 – List of PNFs used in the three UCs of the 5G-MEDIA project .........................134
### Definitions and acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>5G-PPP</td>
<td>5G Infrastructure Public Private Partnership</td>
</tr>
<tr>
<td>AAA</td>
<td>Authentication, Authorization, and Accounting</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ATS</td>
<td>Apache Traffic Server</td>
</tr>
<tr>
<td>BBU</td>
<td>Base Band Unit</td>
</tr>
<tr>
<td>BMS</td>
<td>Building Management Systems</td>
</tr>
<tr>
<td>BSS</td>
<td>Business Support System</td>
</tr>
<tr>
<td>C-RAN</td>
<td>Cloud Radio Access Network</td>
</tr>
<tr>
<td>CSAR</td>
<td>Cloud Service Archive</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Delivery Network</td>
</tr>
<tr>
<td>CESC</td>
<td>Cloud-Enabled Small Cell</td>
</tr>
<tr>
<td>CI</td>
<td>Common Infrastructure</td>
</tr>
<tr>
<td>CI/CD</td>
<td>Continuous Integration and Delivery</td>
</tr>
<tr>
<td>CMU</td>
<td>Classifier and Marker Unit</td>
</tr>
<tr>
<td>CNO</td>
<td>Cognitive Network Optimizer</td>
</tr>
<tr>
<td>CO</td>
<td>Central Offices</td>
</tr>
<tr>
<td>CORD</td>
<td>Central Office Re-architected as a Datacenter</td>
</tr>
<tr>
<td>CoMP</td>
<td>Coordinated Multipoint</td>
</tr>
<tr>
<td>CRAN</td>
<td>Cloud Radio Access Network</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create Read Update Delete</td>
</tr>
<tr>
<td>CSAR</td>
<td>Cloud Service ARchive</td>
</tr>
<tr>
<td>CSE</td>
<td>CogNet Smart Engine</td>
</tr>
<tr>
<td>D2D</td>
<td>Device to Device</td>
</tr>
<tr>
<td>DC</td>
<td>Data Center</td>
</tr>
<tr>
<td>DNL</td>
<td>Data Network Layer</td>
</tr>
<tr>
<td>eMBMS</td>
<td>evolved Multimedia Broadcast Multicast Service</td>
</tr>
<tr>
<td>EC</td>
<td>European Commission</td>
</tr>
<tr>
<td>EMS</td>
<td>Element Management System</td>
</tr>
<tr>
<td>EPC</td>
<td>Evolved Packet Core</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunication Standards Institutes</td>
</tr>
</tbody>
</table>
FaaS  Function as a Service
FSM  Function Specific Manager
GUI  Graphical User Interface
HMN  Heterogeneous Mobile Networks
HTTP  Hypertext Transfer Protocol
IaaS  Infrastructure as a Service
ICT  Information and Communication Technology
InP  Infrastructure Provider
IoT  Internet of Things
IPTV  Internet Protocol Television
KVM  Kernel Virtual Machine
LFC  Lifecycle Control
LTE  Long-Term Evolution
LTE-A  Long-Term Evolution-Advanced
MAC  Medium Access Control
MaaS  Metal as a Service
MANO  Management and Orchestration
MAPE  Monitor-Analyse-Plan-Execute
MDG  Module Development Groups
MEC  Mobile Edge Cloud
ML  Machine Learning
NFV  Network Functions Virtualization
NFVI  Network Functions Virtualization Infrastructure
NFVO  Network Functions Virtualization Orchestrator
NMS  Network Management System
NS  Network Service
NSD  Network Service Descriptors
OAM  Operations Administration and Maintenance
OCP  Open Compute Project
OPNFV  Open Platform for NFV
OSM  Open Source MANO
OSS  Operations Support System
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>OW</td>
<td>OpenWhisk</td>
</tr>
<tr>
<td>OWL</td>
<td>OpenWhisk Light</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof of Concept</td>
</tr>
<tr>
<td>PD</td>
<td>Package Descriptors</td>
</tr>
<tr>
<td>PLS</td>
<td>Physical Layer Security</td>
</tr>
<tr>
<td>PNF</td>
<td>Physical Network Function</td>
</tr>
<tr>
<td>PoP</td>
<td>Point of Presence</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAT</td>
<td>Radio Access Technology</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>RO</td>
<td>Resource Orchestrator</td>
</tr>
<tr>
<td>SDK</td>
<td>Service Development Kit</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Networking</td>
</tr>
<tr>
<td>SFC</td>
<td>Service Function Chain</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SP</td>
<td>Service Platform</td>
</tr>
<tr>
<td>SSM</td>
<td>Service Specific Manager</td>
</tr>
<tr>
<td>SVP</td>
<td>Service Virtualization Platform</td>
</tr>
<tr>
<td>TSC</td>
<td>Technical Steering Committee</td>
</tr>
<tr>
<td>UC</td>
<td>Use Case</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UGC</td>
<td>User Generated Content</td>
</tr>
<tr>
<td>vCDN</td>
<td>virtual Content Delivery Network</td>
</tr>
<tr>
<td>VDU</td>
<td>Virtual Deployment Unit</td>
</tr>
<tr>
<td>VIM</td>
<td>Virtualized Infrastructure Manager</td>
</tr>
<tr>
<td>VLD</td>
<td>Virtual Link Descriptors</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VNFD</td>
<td>Virtual Network Function Descriptors</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
<tr>
<td>VNFFG</td>
<td>Virtual Network Function Forwarding Graph</td>
</tr>
<tr>
<td>VNFM</td>
<td>Virtual Network Functions Manager</td>
</tr>
</tbody>
</table>
Executive summary

The 5G-MEDIA project aims to build an integrated programmable service platform to facilitate the design, development and deployment of media services over 5G infrastructures. To achieve this goal, 5G-MEDIA will capitalize on background elements and previous experiences from 5G-PPP Phase 1 projects, as well as on relevant tools and solutions accessible through the 5G-MEDIA consortium.

The goal of this document is to provide a comprehensive overview of existing tools considered during the design of the 5G-MEDIA platform, explain the rationale behind the selection of specific background elements and present the initial platform architecture design accompanied with sequence diagrams that illustrate the interactions among platform components.

More specifically, Chapter 2 presents the relevant technologies made available to 5G-MEDIA project from 5G-PPP Phase 1 projects, i.e. COGNET, SELFNET, SONATA (also COHERENT, CHARISMA, and SESAME projects are presented in Annex 1) and also the consortium partners. Thus, NFVI OnLife platform implemented by Telefonica, serverless computing OpenWhisk by IBM, monitoring framework from SILO, artificial Intelligence algorithms by UCL, symphony platform by NXW, C-RAN by OTE, and several VNFs for the realization of the use case network services, such as Elastic transcoder and interactive 360° media platform implemented by BitTubes, are presented, as they are being the most influential open source state-of-the-art technologies to drive development in the scope of 5G-MEDIA. In summary, the goal of the second chapter is to provide an overview of the possible exploitable assets accessible by the project and identify the most suitable outcomes from 5G-PPP phase 1 projects, as well as technologies, frameworks and commercial solutions offered by the 5G-MEDIA partners that will be used for the development of the 5G-MEDIA platform.

Chapter 3 focuses on two main core components of the 5G-MEDIA platform, i.e., the Service Development Kit (SDK) that facilitates the development of media services and the Management and Orchestration (MANO) framework used to realize the functionalities of the NFV-MANO stack. For each of these two core components, a description of the candidate solutions coming either from open source projects or from 5G-PPP Phase 1 projects is presented along with an evaluation of the candidate solutions upon different criteria and characteristics. Following the evaluation of the different solutions, the document explains the rationale behind the selection of the Open Source MANO (OSM) for the implementation of the 5G-MEDIA platform.

Having at hand the evaluation and selection of background elements, Chapter 4 presents a high-level architecture design of the 5G-MEDIA platform and explains the main components (i.e., Service Virtualization Platform, SDK) and the extensions with respect to the adopted existing solutions. In addition, Chapter 4 provides sequence diagrams showing the interactions among platform components.

In Chapter 5, the list of the Virtualized Network Functions (VNFs) and the Physical Network Functions (PNFs) to be used in the context of the 5G-MEDIA use cases is presented. The VNFs presented will be proofed, on boarded, deployed and managed through the 5G-MEDIA...
platform to deliver the expected functionalities for each use case and evaluate the 5G-MEDIA solution.

Finally, Chapter 6 presents the key conclusions of this document.
1. Introduction

A great deal of work is currently underway to organize 5G technologies in supporting high-quality streaming media and entertainment applications. In this direction, virtualization and flexible scaling of cloud resources and network services, at both the network’s core and edge, will be the key elements to reduce superfluous operational expenses (OPEX) and lead to shorter time-to-market and lower capital expenditures (CAPEX). The 5G-MEDIA project aims at introducing a programmable edge-to-cloud virtualization framework that exposes several advanced, provable and validated tools and generic networked media services (through APIs, following the microservices based approach, where applicable) to be utilized or combined together to create more complex media application and entertainment services. Based on the open innovation approach of the 5G-MEDIA consortium, the platform will be offered to third parties, targeting mainly Small & Medium Enterprises (SMEs), startups and open source development communities, along with concrete use case scenarios to develop, deploy and validate media and entertainment applications by utilizing the SDK capabilities and service virtualization platform offerings.

1.1. Scope of this deliverable

The scope of this deliverable is to provide the architectural design of the 5G-MEDIA platform and explain the main components of the platform and interactions among them. Since the 5G-MEDIA platform relies on existing background technologies and in this line, it integrates and extends existing components, a primary goal of this document is to provide an overview of candidate background elements considered during the design of the system and explain the rationale for the selection of specific components.

This architecture presented in this deliverable was based on the findings of the deliverable “D2.2 5G-MEDIA Requirements and Use Case Refinement” in order to address the requirements from the different use cases. In addition, an early version of the platform architecture has been provided in “D3.1 Initial Design of the 5G-MEDIA Operations and Configurations Platform”, which focused on the specification of the 5G-MEDIA service virtualization platform. In this deliverable, we have revised this initial high-level architecture to depict more precisely the design of the 5G-MEDIA solution. Concurrently to D2.3, the 5G-MEDIA project releases also the following technical deliverables of WP3-WP5:

- D4.1: 5G-MEDIA Catalogue APIs and Network Apps.
- D5.1: 5G-MEDIA Programming Tools.

Thus, technical specifications and detailed aspects of implementation of the various (sub)components and services (e.g. description of data models, Application Programming Interfaces (APIs) endpoints etc.) of the 5G-MEDIA architecture are not included in this deliverable but in the corresponding technical ones of WP3-WP5. Yet, the results herein are meant to be a reference and coordination point for the implementation and prototype integration activities carried out in these WPs. Further refinements and fine-tunings of the
platform architecture and specification along the project will be documented in the deliverable “D2.4: Final Report on Architecture, Requirements and Specification”.
2. Core Technology and State-of-the-Art

In this section, we introduce several ICT, networking and virtualization technologies and software solutions which are used in the scope of 5G-MEDIA project. Note that the description is not exhaustive, and the intention is just to highlight and briefly discuss a number of exploitable state-of-the-art offerings which could provide tangible benefits for the 5G-MEDIA project, under the proper adjustment and/or extension. We especially focus on two sources of transfer knowledge; the first includes subsystems and offerings of 5G-PPP phase 1 projects that could provide the baseline to deliver certain components of the 5G-MEDIA software architecture, while the second includes numerous commercial tools, services and technologies brought by 5G-MEDIA partners to enrich considered use cases and improve provided services.

2.1. Projects and offerings of 5G-PPP phase 1

The 5G-MEDIA consortium has extensively evaluated the 5G-PPP Phase 1 projects, particularly those where 5G-MEDIA partners have been involved, to identify outcomes and offerings that would be exploited and further enhanced in its scope (also based on the individual partners’ exploitation plans). In the following subsections, a brief overview is provided for the 5G-PPP Phase 1 projects COGNET, SELFNET, and SONATA. These projects offer several potentially exploitable innovations and technologies which are aligned with the use cases and the system requirements set by the 5G-MEDIA project. Thus, apart from their technical concept and innovations, their relevance and potential exploitation in the scope of 5G-MEDIA is also discussed.

In Annex I (Section 7), also the main outcomes and key aspects of the projects COHERENT, CHARISMA, and SESAME are presented. Even if it was finally decided by the 5G-MEDIA consortium not to directly exploit this legacy in the initial design of 5G-MEDIA architecture, they provide numerous advances and results that could be potentially used in a later phase of the project.

2.1.1. The project COGNET

2.1.1.1. Overview

The project COGNET (labelled as CogNet) aims at developing solutions that will provide a higher and more intelligent level of monitoring and management of networks and applications, improve operational efficiencies and facilitate the requirements of the 5G. CogNet aims to make a major contribution towards autonomic management of telecoms network infrastructure by investigating existing and devising novel machine learning (ML) algorithms tuned for available network data in order to yield insights, detect meaningful events and conditions and respond correctly to them. CogNet has produced a general architecture and its associated information model, based on the challenges of future 5G network management, such as network resource utilization, network performance degradation, and energy efficiency. The use cases considered in CogNet are related to:
• Situational Context, presenting how the system will handle exceptional situations due to external environmental conditions which cannot be directly detected within telecommunication systems.

• Just-in-time Services, referring to how cognitive network management techniques will enable the reduction of creation and deployment time for network services in 5G.

• User-Centric Services, moving towards a richer and more complex service catalogue, with the capacity of tailoring services to the particular user’s needs.

• Optimized Services in Dynamic Environments: enabling the network to be deployed, scaled and migrated with ease and speed unheard of in today’s networks, specifically by relying on the virtualization of the network functions.

• Service Level Agreement (SLA) Enforcement, handling in an automated and efficient way the level of service guaranteed to a user or service by the network operator.

• Collaborative Resource Management, where both the network and the applications at both endpoints exchange metadata about the network flows to improve network conditions and user experience.

![Figure 1 – CogNet solution](image-url)

The CogNet architecture, shown in Figure 1, complements the NFV reference architectural framework produced by ETSI. The state and consumption records on the hardware resources are gathered in real-time from multiple functional blocks constituting the layered architecture. The collected records will be processed by the CogNet Smart Engine (CSE) periodically or in (near) real-time, to create insights from telecom data or to recommend policies best matching

---

the network management goals. Real-time analysis is one of the core contributions of this work. Such a capability is crucial to 5G network management since it aims to provide immediate response to changes.

A first prototype of the ML components has been implemented as part of the CSE that includes the following capabilities/tools:

- Automated machine learning model selection (TCDC).
- Feature selection (PICS, IterFS).
- Supervised (Spark-TK, NetSpark, LSSVM-SPARK) and unsupervised (PSCEG, Funco) machine learning tools.
- Streaming-mode classification (streamCluster) and regression (Spark - streaming - linear - regression).
- Heuristic function optimization network management support (SAOptimizer).
- User throughput prediction (ML4MQ).

The most salient characteristic of the CogNet architecture is the creation of a double closed control loop with common interfaces in each one. One control loop is the “classical” one, where CogNet proposes to apply cognitive control to an integrated NFV/SDN Software Network architecture, which provides the data to be fed into the cognitive control module. The other control loop is the machine-learning one, where different algorithms and modules can be applied in order to shape (and control) the cognitive controller of the first loop, using a set of data derived from the same sources as the classical loop. The CSE, acting as cognitive controller, at the top of one loop and the bottom of the other, is the nexus between them.

The general implementation of the CogNet double control loop includes data capturing, data processing and decision recommendation. CogNet has implemented a Common Infrastructure (CI) based on open-source components for all the different deployments to be analysed and optimized. This CI brings the following objectives:

- Replicability: the solution can be instantiated, and the same experiment can be executed over different platforms/test-beds.
- Integration: a common set of elements, dataflow, SW stack and APIs to be considered and adopted across all the experiments.
- Pedagogical: provide a best-practice reference platform for developing cognitive management solutions with machine-learning.

The CI is a set of resources that enables experimentation over a common testbed. Each demonstrator meets different use cases and scenarios, eventually need third party components or just the core ones and employs alternative technologies to build the required environments.

Each system can be mapped to a demonstrator bringing a set of ML algorithms tuned to intrinsic features from the data and with a range of possible actions to be triggered accordingly. Moreover, the data ingestion and the policy recommendation are embodied in two APIs that decouple the CogNet control loop from the analysed and optimized system to be managed.
2.1.1.2. Technical approach

CogNet architecture can dynamically adapt to changes in the network by combining machine learning solutions. For develop this adaptation, it performs a Monitor-Analyse-Plan-Execute (MAPE) autonomic loop with the following main responsibilities:

1. Monitor: Data collection and monitoring to gather details from the managed computing resources and network environments.
2. Analyse: The information is forwarded to an analysis engine to assist the management of the network.
3. Plan: The output of the analysis engine generates network policies stating concrete actions over the network infrastructure.
4. Execute: Planned actions are performed over network resources, e.g. Virtual Network Functions (VNFs), Network Functions Virtualization Infrastructure (NFVIs) etc., through the Management and Orchestration (MANO) controller.

Figure 2 shows the logical components of CogNet platform, with the applicable technology stacks.

The major components of the architecture are:

- The CogNet Smart Engine (CSE): CSE receives records about network state and consumption, and then, it applies Deep Neural Network algorithms and other ML algorithms to automatically extract informative features from the records. It can identify the Machine Learning models that will be deployed on the engines comprised on the CSE, either in an autonomous mode or based on customer requirements. Once the models are selected and the features are extracted, one of the two possible engines is used, particularly:
• The Batch Processing Engine: it trains the model and generates scores. It also evaluates the performance of the model on the previous iteration of the MAPE loop, if it is stale, it generates a new model. This engine works in a more accurate but higher latency manner.

• The (near) Real-time Processing Engine: it applies the model trained at Batch Processing Engine to obtain scores in a low-latency mode. It can also train a model directly by using some on-line clustering algorithms. A combination of both engines may add complexity to scoring process in order to apply some post-processing. Finally, CSE returns some thresholds or metrics to the Policy Engine. Another version of CSE is the Light-weight CSE (LCSE) which only contains the (Near) Real-time Processing Engine and works in low-latency mode, in a less accurate way. The LCSE, is included on the MANO stack.

• The Policy Engine: it is mainly responsible for mapping the thresholds and metrics given by (L)CSE into policy actions which are understood by MANO stack. Considering these recommendations and the state of individual network elements, it matches them with a specific policy, specifying how to manage network resources for a required service. It can dynamically respond to different situations and changes on network environment. It can also adapt or create new policies, based on the experience of previous policies.

• The Network Function Virtualization (NFV) Architectural Framework: it leverages the ETSI NFV architecture. A key innovation is to adopt intelligence to MANO which is enhanced with a LCSE, for working as a CSE but in low latency mode, and a Proxy, which forwards the policy actions from Policy Engine to MANO stack, translating these actions into a format understandable by MANO stack components.

In Figure 2, system X represents a VNF set, to be analysed and potentially optimized, that processes different traffic with challenges to be met. Monasca² collects the metrics from the System X and stores them into time series in an InfluxDB database. At the same time, Monasca publishes the metrics through a live channel based on Kafka. For those systems requesting data on demand InfluxDB provides the stored records. Then, Machine Learning algorithms, encapsulated on Dockers, process the metrics, from InfluxDB or the live Kafka topic, and produce events with the results. These events are sent to the Policy Engine through Kafka. From there, the Policy Engine checks the different Policies defined in a SUPA ECA format³. According to the satisfied conditions, from the policies, some specific actions could be triggered to System X.

Each system, as a set of VNFs, can be mapped to a demonstrator, which exercises the CogNet CI, bringing a set of ML algorithms tuned to intrinsic features and with a range of possible actions to be triggered accordingly. This does not imply a 1-to-1 relation for algorithms and actions. In addition to the Common Infrastructure deployment, the CogNet project has

---

² http://monasca.io/

struggled with the lack of usable data sets since it started, thus it developed the concept of an open environment for the generation of significant synthetic datasets, suitable to be applied to the different use cases in the project and generalized to further application of data-driven network management. To produce the datasets, the CogNet has analyzed several open formats and open source tools for the generation of network flow information, and finally selected tstat\textsuperscript{4}.

2.1.1.3. Innovation

The main innovation of CogNet project is the CSE which adds intelligence to the management of the network. This element monitors the network by using Deep Learning techniques to know which features are relevant for the management, and after that, it can dynamically adapt to the state of the network and its environment by selecting different Machine Learning models. CSE is enhanced by two engines:

- **Batch Processing Engine**: for process data in high-latency mode with high accuracy. It trains the selected Machine Learning model and score the data. It can also create a new Machine Learning model if necessary.

- **(Near) Real-Time Processing Engine**: for process data in low-latency mode with less accuracy. It applies a Machine Learning trained by Batch Processing Engine for scoring data. With on-line clustering algorithms, it can also train a model directly.

LCSE, i.e. the lighter version of the CSE (LCSE), is included inside the MANO stack, so the last can execute the policies in low-latency mode. LCSE version only contains the (Near) Real-Time Processing Engine and it is located as close to data as possible in order to reduce and processing time.

2.1.1.4. Relevance and potential exploitation in 5G-MEDIA concept

The 5G-MEDIA project leverages on the cognitive control technology (e.g. CSE) and ML algorithms of CogNet to implement the Cognitive Network Optimizer (CNO), a core component of 5G-MEDIA software architecture, which is responsible to dynamically establish and update the VNF Forwarding Graphs (VNFFGs) of the Network Services (NSs) instantiated under the control of its Service Virtualization Platform (SVP). Also, an effort has been made in order to use similar tools throughout the MAPE cycle, although this was not possible in all cases either due to the technological advancements that have taken place between the time of execution of the two projects or due to operational characteristics that were not covered by the technologies used by COGNET. For example, the use of time-series based database, i.e. InfluxDB, by COGNET has been used in the first release of the 5G-MEDIA platform, but the final goal of the project is to capitalize on big data-based databases that satisfy the requirements of scalability. Moreover, Monasca was an active OpenStack project by the time of COGNET project execution, but the decision towards monitoring OpenStack infrastructure in 5G-MEDIA includes Gnocchi, firstly due to the fact that it is more advanced that Monasca and secondly

\textsuperscript{4} http://tstat.polito.it/
because the consortium wanted to be as close as possible to solutions utilized by OSM. On the other hand, Apache Kafka has been selected in both projects although the use of RabbitMQ has been considered. With respect to the Analysis tools, Apache Spark was the selected platform for both projects, while 5G-MEDIA is in parallel moving towards integrating some of its AI algorithms in TensorFlow, being another state-of-the-art tool nowadays.

2.1.2. The project SELFNET

2.1.2.1. Overview

The 5GPPP Phase 1 SELFNET project is realizing an autonomic network management framework for software-defined 5G networks capable of:

1. Automated deployment of network management tools and functions.
2. Automated network monitoring and autonomic network maintenance via high-level tactical measures.
3. Autonomic corrective and preventive actions to mitigate existing or potential network problems.
4. Deploying tools and mechanisms to protect the network from various threats.

In SELFNET, VNFs implement either control or data plane functionalities (i.e. sensor and actuators) on top of a virtualized infrastructure managed by a cloud management system. Software Defined Networking (SDN) applications implement control plane functions needed to interconnect VNFs and can even possibly run directly on top of a physical infrastructure. The project is working on enhancements to SDN controller OpenDaylight\(^5\) and ETSI NFV MANO tools (e.g. Open-Baton\(^6\)) to obtain a reference 5G control and management network architecture. SELFNET is using the Metal as a Service (MaaS) tool to manage a pool of infrastructure nodes and realize a portable testbed.

2.1.2.2. Technical approach

The SELFNET system architecture is depicted in Figure 3.

\(^5\) https://www.opendaylight.org/
\(^6\) https://openbaton.github.io/
Starting from the bottom of the stack, the Infrastructure Layer contains the Physical Sublayer and the Virtualization Sublayer. The Physical Sublayer contains all the physical elements of the network, as well as the physical servers available on the Data Center (DC). On top of the Physical Sublayer is provided the Virtualization Sublayer which provides access to the virtual resources of the DC (compute, storage and network) through a hypervisor. The Virtualization Sublayer represents the Network Functions Virtualization Infrastructure (NFVI)\(^7\).

On top of the Infrastructure Layer is located the Data Network Layer (DNL), which represents an explicit architectural evolution towards the SDN paradigm. Several types of network functions are part of the DNL, as presented in Figure 4. A network function can be either a non-virtualized network function or a VNF. When virtualized, the most common model, at least up to now, is to include both control and data plane functionalities, which is not fully compliant with the SDN paradigm since it does not allow the centralized control of its data plane functions.

\(^7\) http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf
On top of the DNL is the Control Layer, which includes two internal sublayers: SDN Controllers Sublayer and the SON Control Plane Sublayer. The SDN Controllers Sublayer comprises a group of horizontal and vertically distributed SDN Controllers, whereas the SON Control Plane Sublayer represents the network functions control plane, being either actuators or sensors.

On the right side of the SELFNET architecture diagram is the NFV Orchestration and Management Layer (OML). It corresponds to the ETSI NFV Management and Orchestration (MANO) layer and is responsible for orchestrating and managing the whole set of virtual functions that are embedded on the SON Control Plane Sublayer and on the SON Data Plane Sublayer. As sublayers, it includes (partially) the Orchestration Sublayer, the VNF Management Sublayer and the Virtualized Infrastructure Manager (VIM) Sublayer. The Orchestration sublayer part in the OML corresponds to the ETSI MANO Network Functions Virtualized Orchestrator (NFVO) and is responsible for orchestrating the virtual resources and network functions. The VNF Management and the VIM sublayers, are responsible for the VNFs and virtual resources management, respectively. These sublayers also correspond to the ETSI MANO VNFM and VIM, respectively.

On top of the already described layers and sublayers are the SON Autonomic Layer and the SON Access Layer. The SON Autonomic Layer provides the SON intelligent mechanisms, namely:

- Network sensing and SON indicators production – Monitor & Analyzer Sublayer;
- Diagnose the network condition and define the set of corrective actions – Autonomic Management Sublayer;
● Organized enforcement of the corrective actions on the network (physical/legacy and/or virtual) – Orchestration Sublayer;
● Provide new network functions to the SELFNET architecture – VNF Onboarding Sublayer;

The topmost layer is the SON Access Layer, which encompasses the interface functions that are exposed by the framework. Despite the fact that internal components may have specific interfaces for the particular scope of their functions, these components contribute to a general SON API, managed by the SELFNET API Broker Sublayer, that exposes all aspects of the autonomic framework to external actors, e.g. Business Support Systems (BSSs), Operational Support Systems (OSSs) and Administration Graphical User Interface (GUI). The GUI provides the network administrator the capability to interact with and configure the framework components (e.g. stop, verify or manually enforce any of the actions that SELFNET is governing) and also obtain the complete status of the network.

2.1.2.3. Innovation

The SELFNET project introduces the following major innovations:

- An autonomic network management framework, based on integrated software technologies and intelligence, which reduces CAPEX/OPEX.
- A software-based automation of services deployment and 5G physical and virtual MEC infrastructure, reducing service creation and deployment time.
- A cross-layer, multi-tenant-aware monitoring system, enabling a holistic view of software networking based 5G networks.
- The enabling of Health of Network use cases, improving reliability, security and QoE.

2.1.2.4. Relevance and potential exploitation in 5G-MEDIA concept

Results from SELFNET on the ETSI NFV MANO, in particular the NFV catalogue mechanisms and the NFV MANO layer for VNF management and NSs orchestration, are useful references and assets that have been selected for utilization in 5G-MEDIA to build the use case scenarios and preliminary instantiations of the 5G-MEDIA orchestration and management layer, combined with the NFV catalogue mechanisms. NXW is a partner in SELFNET consortium and brings open source background to 5G-MEDIA as well as opportunities for collaborations between the two projects.

Specification and prototype implementation of mechanisms and frameworks for the seamless management and orchestration of VNFs, the virtual network instantiation and monitoring services, the virtual network and flow re-planning.

In SELFNET, NXW has extended Open-Baton, developing a dedicated VNFM, but most importantly introduced an enhanced Catalogue that will be used by 5G-MEDIA as the starting point for the implementation of a cross-MANO solution. In particular, NXW has led the development on NFV and SDN APP Repository & onboarding service. In particular, the App repository is new software package that has been integrated with Open-Baton and will be extended in 5G-MEDIA to be integrated with OSM with respect to onboarding. The ETSI MANO
VNF onboarding procedures has been adopted and has been enhanced in terms of information models and mechanisms to accommodate SDN applications and specific SELFNET autonomic management requirements.

2.1.3. The project SONATA

2.1.3.1. Overview

SONATA addresses the significant challenges associated with both SDN and NFV by the development and deployment of an orchestration platform for the complex services envisioned for 5G networks and empowered by these technologies. In particular, SONATA has developed an NFV framework that provides a programming model and development toolchain for virtualized services, fully integrated with a DevOps-enabled service platform and orchestration system. The project is based on the following three high-level objectives:

1. Reduce time-to-market of networked services, by streamlining development with abstract programming models, SDK and a DevOps paradigm that integrates operators, manufacturers and third-party developers.
2. Optimize resources and reduce costs of service deployment and operation, by orchestrating complex services to connectivity, computing and storage resources, and automatically re-configures running services.
3. Accelerate industry adoption of software networks and support the full-service lifecycle: development, testing, orchestration, deployment, management and operations, defining a roadmap for uptake of its results towards stakeholders’ larger transition to SDN/NFV.

Out of the six use cases envisioned as possible domains that SONATA outcomes will have a strong impact in the beginning of the project, i.e. Internet of Things, virtual Content Delivery Network, industrial networks, virtual Evolved Packet Core, Personal Security Service, Separation of client and hosting Service Providers, three of them have become demonstrable Pilots during the project lifetime. However, those cover all the SONATA validation aspects from the validation of functional requirements to non-functional, plus verify the initial promise to provide a CI/CD framework for Telecom Operators to support multiple verticals and business scenarios. In more details:

- Virtual Content Delivery Network (CDN) Pilot: Two scenarios are anticipated for this pilot, namely:

---

8 For further information, please refer to:
https://bscw.selfnet-5g.eu/pub/bscw.cgi/d18751-4/*/*/*/*/DOI-D2.1.html
https://bscw.selfnet-5g.eu/pub/bscw.cgi/d37341-4/*/*/*/*/DOI-D2.2.html
https://bscw.selfnet-5g.eu/pub/bscw.cgi/d37388-4/*/*/*/*/DOI-D2.3.html
https://bscw.selfnet-5g.eu/pub/bscw.cgi/d37407-4/*/*/*/*/DOI-D2.4.html
• Classic vCDN mode: Content originates from a single content provider or multiple ones, distributed across the vCaches and eventually delivered to a huge number of subscribers. This scenario will be used to highlight placement and scaling functionalities of the SONATA service platform.

• User Generated Content (UGC) based vCDN mode: In this scenario, content originates from the end users, allowing various sub-cases of social networking content exchange. The SONATA service platform allows the flexibility of dynamically extending the vCDN service, accommodating additional sources from alternative Content Providers. The twist of this scenario is that the UGC content is identified and cached at the edges, allowing resource optimisation at the edges. This scenario, reveals the interaction of the service platform with information that stems from the network (either as traffic information or content information or end-user information) in order to dynamically configure and optimise the CDN for an improved user experience.

• Personal Security Application: This pilot focuses on showcasing and assessing the SONATA system capabilities in order to enhance a service provider based personal security application. To this end, a security application comprising several different security components such as a firewall, a virtual private network service and an intrusion detection system, is executed in the virtual network infrastructure of the service provider. It is embedded in the data path of a user and assesses and filters its network traffic and thus protects its devices connected to the Internet. Using a self-service portal, a user can connect to the personal security application and adapt the actual composition of the network functions that constitute the service. Thus, a user might add a firewall or an intrusion detection system to its data path on demand.

• Hierarchical Service Providers: The goal of this deployment scenario is to showcase the deployment of a network service in a recursive SONATA service platform environment. To do so, two full service platforms, each having its own NFVI infrastructure available, are deployed and connected in a recursive manner. The first platform, referred as upper-SP, is the entry point for the service instantiation requests done by the BSS. The second platform, referred as lower-SP, has its northbound interface connected to the southbound interfaces of the upper-SP to which it offers service deployment capabilities. The network service used for this scenario is a simplified version of the network service used in the vCDN pilot. Like in the vCDN pilot, the service is deployed between a content provider and the end users.

2.1.3.2. Technical approach

The software architecture of SONATA service platform is shown in Figure 5. The first major component is an SDK that supports service developers with both a programming model and a set of software tools. The SDK allows developers to define complex services consisting of multiple VNFs. A service provider (which can also be the service developer) can then deploy and manage the created services on one or more SONATA service platforms through the corresponding gatekeeper components. Services and their components can also be published in catalogues to be reused by other service developers and providers.
SONATA’s flexible service platform is the second major component of the system. Due to the fully customisable and modular design of its MANO framework, the service platform offers customisation opportunities on two levels. First, the service platform operator can modify the platform, e.g., to support a desired business model, by replacing components of the loosely coupled MANO framework (MANO plugins). Second, service developers can influence the orchestration and management functionalities of the platform pertaining to their own services, by bundling small management programs, so-called function- and service-specific managers (FSMs/SSMs), with their services. This enables a new level of service control capabilities for service developers such as influencing placement decisions of services deployed across multiple points of presence (PoP). These PoPs are cloud DCs operated by infrastructure operators on which SONATA controlled services are executed. The SONATA system is designed for agile development and operation of network services. It enables a DevOps workflow by virtue of the full compatibility and integration between the SDK tools and the service platform. In this way, developers and operators can collaborate on design, development, deployment, and lifecycle management of network services, as well as optimise and adapt the design and implementation of the services based on collected monitoring information regarding the current state of the service and network resources.

2.1.3.3. Innovation

Through a customisable service platform with NFV Orchestrator supporting Network Service SDK for developers and specialised DevOps workflow to connect it all, SONATA pushes the following innovation towards the transition to software networks.

- Modular and Customisable MANO Plug-in Architecture:
  - Providing NFV MANO flexibility to network operators with customisable platform functionality and ability to add new features via plug-ins (FSM).
  - Empowering third-party service developers with control over specific orchestration and management functionalities pertaining to their own service (SSM).
• Supporting both resource and service orchestration. Advanced conflict resolution for resource allocation introduces an auction-based mechanism to establish priorities among services.

• Interoperable and Vendor Agnostic Framework:
  • Multi-VIM, multi-vendor, multi-site support.
  • Underlying ETSI-based architecture.
  • SDN/NFV integration for better interoperability between NFV MANO layer and SDN controller.

• Efficient Network Service Development and NFV DevOps:
  • Providing service developers with an SDK for efficient creation, deployment and management of VNF-based network services on the platform.
  • Unique NFV DevOps workflow and platform/SDK support bridges collaboration between operators and service developers.

• 5G Slicing and Recursion Support:
  • Slicing support delivers performance isolation and bespoke network configuration for industry verticals foreseen in 5G networks.
  • Recursion support allows stacked tenant and wholesale deployments in new software networks business models.

2.1.3.4. Relevance and potential exploitation in 5G-MEDIA concept

The SDK of SONATA has been used as the baseline to design and implement the SDK of the 5G-MEDIA project. In SONATA, the SDK is composed of the following main components:

• A workspace and packaging library to set up the workflow and the environment of service development in line with the basic SONATA’s service programming model.

• A text-based editor allowing the service developer to read SDK schemata, integrate descriptions of VNFs/NSs, interact with service logic modules from internal and external catalogues and access service debugging and profiling tools (e.g., by highlighting measured VFNs and VNF connections).

• A separate internal catalogue environment containing the package descriptors (PD), network service descriptors (NSD), VNF descriptors (VNFD), etc. that are either imported from other third party or developed by the service developer itself. The main objective of the SDK catalogues is to support the development of new NSDs, VNFDs, etc. by making existing NSDs, VNFDs, etc., developed by other developers, available locally.

• A tool for packaging components of a service before delivering the service to an operator.

• An emulation platform to support network service developers to locally prototype and test complete network services in realistic end-to-end multi-PoP scenarios.

SONATA monitoring framework is another outcome of SONATA project that has been partially exploited in 5G-MEDIA. The monitoring framework developed in SONATA leverages on state-
of-the-art monitoring & messaging tools (e.g. Prometheus monitoring system\(^9\)) to measure network, compute and storage metrics of instantiated VNFs/NSs and store them in an isolated way, which can be accessed by any internal component of the SVP and be linked to any high-level service-oriented viewpoint provided by the SDK. This way, the monitoring framework can provide QoS tracking and VNFs overview services both in runtime and service development periods. In the SONATA SDK, the monitoring tools are able to support a wide range of debugging and profiling features at the disposal of a service developer.

### 2.2. Tools and solutions transferred to 5G-MEDIA by partners

In this subsection, an overview is provided of technologies, tools and research/commercial solutions developed or supported by the consortium members of 5G-MEDIA project that can be exploited and extended in the scope of project’s use cases and scenarios. For each technology/tool, its main innovations and key characteristics are firstly presented and its potential relevance to the 5G-MEDIA is later discussed.

#### 2.2.1. OnLife platform

**2.2.1.1. Overview**

Traditionally, Telecommunications Central Offices (CO) or Telephone Exchanges require significant efforts for configuration and control. Meanwhile, users continuously demand higher capacities, lower latencies and more customized services. The OnLife project, inspired by the Central Office Re-architected as a Datacenter (CORD\(^10\)) initiative, plans on breaking this paradigm, bringing the benefits of cloud computing to the Central Offices. Accordingly, it introduces “Central Telefonica de procesamiento de datos (CTpd)”, a new architecture for Central Offices based on edge computing that allows the virtualization of its access network and offers third-party application developers and content providers cloud-computing capabilities at the edge of the network. The CTpd proposes an innovative design based on NFV, SDN and cloud computing paradigms, designed within the OnLife Innovation project. CTpd builds on some CORD principles, taking its disruptive approach a step further by simplifying the implementation and introducing new elements, including a native IPv6-only fabric and an innovative rack design for its hardware infrastructure.

OnLife is an innovation project, whose main goal is to design a future-proof technology stack that could bring the benefits of cloud computing and software (network programmability) to the access network in the CO. Therefore, the CTpd solution must not only be able to support current residential services, namely Internet access, voice calls and Internet Protocol Television (IPTV), but also new provider services and third-party solutions that could smoothly be deployed on a cloud environment, built with commodity infrastructure. Additionally, there were three main principles that guided the CTpd design, namely:

\(^9\) [https://prometheus.io/](https://prometheus.io/)

1. Greenfield: No support/integration with legacy systems nor infrastructure,
2. Extreme use of simplification and NFV principles: Making NFVs as simple as possible and only implementing the protocols and pieces of CORD that were necessary for the actual services to be provided, but leaving room to increase functionality in the future if needed, and,
3. Open Source Software and Open Compute Project (OCP) hardware.

Aiming to a future-proof solution and aligned with the Greenfield principle, CTpd only supports IPv6, and IPv4 is considered a service with interconnection to IPv4 Internet provided by means of NAT64 at the edge of the data center or tunneling the IPv4 customer traffic to that point. Further, following the simplification principle, CORD is implemented using a different platform for cloud computing, i.e. OpenStack and XOS are replaced with OpenNebula, which is a lightweight and flexible solution to enable fast prototyping, service orchestration and management of the CTpd infrastructure. Several CORD modules, such as Authentication, Authorization, and Accounting (AAA) or Virtual Tenant Networks (VTN) are not implemented, as its functionality is redundant to other elements to be deployed in CTpd or not required to provide the selected services. Thus, OpenNebula represents a clear advantage in this area compared with other approaches, mainly for the following reasons:

- Smaller footprint, in terms of hardware requirements, software components and operational costs.
- Simple and extensible architecture, with well-defined interfaces that allow an easy integration of new and innovative scenarios.
- Rich and consistent feature set, to support novel networking services.

Finally, the characteristics of OpenNebula also allow the CTpd to design a network service development framework. This framework will open the CO to third-party applications to explore new business use cases.

The architecture of CTpd bases on CORD, but it has been modified following the design principles described in the previous section. Figure 6 illustrates the architecture of the CTpd project. As shown, upon OpenNebula, ONOS is the selected SDN platform for the deployment of CTpd, as most of CORD functionality relies on applications developed in this platform.
One of the main challenges in CTpd is to open the CO to third-party edge computing applications, similarly to the Infrastructure as a Service (IaaS) model, which opens the data center to external workloads. The ability to provide this edge computing platform in a pay-as-you-go model, a la IaaS, opens up avenues in both innovative use cases and business models. However, given the specific characteristics of the CO in terms of computational and storage resources, and the security constraints of the environment, it is required a well-defined framework to develop such edge applications. In CTpd an edge computing application has the following characteristics:

- **Stateless:** In order to fast reallocate an application, or to migrate it when the user moves across the access network (e.g. from home to the office), the edge application cannot store state within the CO. Any state persistence cannot be stored at the edge, so it is proposed to be stored within the BSS through well-defined interfaces in dedicated storage services.

- **Autoconfiguration:** The application should be able to autoconfigure itself. This process is performed using specific information (context) passed to the edge application upon boot. The context may include user data, configuration parameters or additional resources to install the application. In this phase, the edge application will retrieve any state data needed from the BSS storage services.

- **Composition:** There are some complex applications that require the deployment of multiple VMs. An edge application in CTpd captures this nature and includes also deployment dependencies between the VMs. The inter-connection of the VMs of each edge application happens in a separate private network.

- **Elasticity:** Considering application specific performance metrics, CTpd can increase (or decrease) the number of VMs or application components. An elasticity rule may require for example to add more VMs at specific times and dates (e.g. with an advertisement campaign) or when the number of requests are above a given threshold.
Apart from the above characteristics, CTpd also provides a well-defined API to manage the edge application. This API resembles the classical IaaS API to control the life-cycle of a VM. CTpd uses the functionality exposed by OpenNebula and ONOS to deploy the edge application and provide it with the features mentioned above.

The implemented Proof-of-Concept (PoC) of this architecture consists of three main components:

- **Hardware**: Network devices and VMs are virtualized in two different servers with 8 CPUs, 256GB HDDs, and 16GB and 32GB of RAM memory, respectively.
- **Software**: All network applications are similar to those in CORD but have been redesigned to work with the CTpd architecture (more information about this is provided in the next subsections).
- **Management**: Service management is coordinated by OneFlow, which relies on OpenNebula and ONOS for infrastructure and network management.

The use of real (non-emulated) hardware is a key component, which will be implemented in the next phase of the project. The main element of the hardware design is an OCP-based rack, which will be also equipped with OCP hardware: vOLTs, vROADMs, switches and servers (computing infrastructure). One key aspect of the design is that the chosen rack follows the “48V Open Rack v2.0” specification, which is widely available in COs, as -48V is still used to power the Public Switched Telephone Network.

The main differences between CORD and the CTpd implementation are:

- CTpd implements IPv6 natively, whereas the current CORD reference implementation supports only IPv4.
- CTpd uses Business Support System (BSS) to authenticate the users by means of an authentication portal in the PoC implementation, while CORD uses 802.1x, which is not supported in the targeted access networks, and an AAA application to perform the same function.
- CTpd replaces the vRouter functionality with a custom routing management application, called vErC. The vRouter application is not used as it does not fully support the IPv6 functionalities required by CTpd.
- The vOLT application, which manages the OLT hardware, has been adapted for the PoC as a reactive behavior is required to enable automatic provision of new customers.
- The VTN application is replaced in CTpd with a simplified forwarding application, ClosFwd. This new application manages traffic flows within the CTpd.
- CORD uses virtual Subscriber Gateway (vSG) as a virtual replacement of the Customer Premises Equipment (CPE) installed in containers. The current PoC of CTpd replaces this element with a new one (vPdC), which provides a simplified functionality enabled by its IPv6-only design and running on a VM.
- CORD uses XOS, while CTpd relies on OpenNebula OneFlow and its interaction with the BSS to orchestrate the services.

Resource monitoring and management in the CTpd PoC was implemented with Iris, a cloud-based tool that provides Data Analytics as a Service (DAaaS), enabling collection of data. This
way, logs from the different software applications are collected and relayed to a VM that preprocesses them before uploading them to Iris, where real-time data is accessible using Kibana dashboards.

As an example of the previous application, a CDN service has been defined and deployed at the edge. This scenario is represented in Figure 7. The system automatically deploys the DNS and content nodes in CTpd. The application is prepared to automatically install and configure itself with the information provided in the context. Finally, a service VLAN is assigned to the CDN virtual machines and routes for the CDN traffic are installed in the Clos to connect the vPdC with the CDN service. Therefore, the traffic flow would start at the client’s residence, then to the vPdC traversing the vOLT and the Clos fabric, and it would end at the CDN after traveling through the fabric again. Instead of the CDN, the ErC—which routes packets to the Internet– or other services (serv1, serv2) could be the destination of the traffic.

![Figure 7 – Service architecture of the CTpd project](image)

2.2.1.2. Relevance to 5G-MEDIA use cases and platform architecture

5G-MEDIA has achieved the integration of OnLife infrastructure and CTpd virtualization solution with its SVP towards using it as an NFVI to demonstrate broadcasting scenarios in Use
Case (UC) 2 “Mobile Contribution, Remote and Smart Production in Broadcasting” and being an important exploitation and marketing asset for Telefonica, as the leader of European OSM solution, towards extending the list of supported NFVIs, fully integrated to OSM, in the race of competition with ONAP and OPNFV, being the selected MANO frameworks in China and U.S.A.

2.2.2. C-RAN

2.2.2.1. Overview

Radio Access Network (RAN) architecture\(^\text{11}\) typically appears under two different approaches: the distributed and the centralized. The legacy RAN architecture is distributed in which the complete baseband deployment (including all protocol layers) is located in each cell separately, while on the other hand Cloud-RAN\(^\text{12}\), or Centralized-RAN (C-RAN), is based in a centralized topology, where the baseband deployment serves several cells.

The increasing complexity and traffic in new generation LTE networks make C-RAN architecture more and more attractive to telecommunication providers. In Figure 8, the C-RAN architecture is presented compared to the legacy RAN architecture. C-RAN architecture offers innovative capabilities, such as virtualization, centralization of hardware and coordination between cells. For instance, it becomes easier to upgrade or troubleshoot the network when necessary as well as the same hardware could be shared by many cells since the topology is centralized. Additionally, all the cells could fully coordinate with each other, since there is a central management, avoiding overlays and waste of resources. At the same time, the ability to use NFVI could increase network flexibility.

\(^{11}\)https://www.ericsson.com/assets/local/publications/conference-papers/5g_radio_access_network_architecture.pdf

C-RAN architecture could also be divided into two types: the one with fixed BaseBand Units (BBUs) and the one with virtual BBUs. In Figure 9, the differences between the 3 types of RAN (traditional, with fixed BBUs and with virtual BBUs) are presented in Pareto Front.

Figure 8 – left) Legacy RAN architecture, right) C-RAN architecture

Figure 9 – Pareto Front of different converged architectures

2.2.2.2. Relevance to 5G-MEDIA use cases and platform architecture

OTE already uses IP RAN architecture to provide voice and Wi-Fi access. The existing RAN
architecture is presented in Figure 10. The improvement of offering services becomes obvious by comparing three key indicators between the older network architecture and the IP RAN architecture:

- Interrupted voice calls are decreased by 15%.
- Invalid voice calls are declined by 27%.
- Failure to access HSDPA for data transfer is limited by 56%.

**Figure 10 – OTEs IP RAN architecture**

C-RAN architecture could offer many advantages in 5G-MEDIA project implementation. In an ecosystem where the network applications are rapidly developed and instantiated at the edge cell, it could enhance management KPIs by making network management centralized and facilitating the enrichment of provided services with virtualized ones. To this scope, C-RAN functionality could be orchestrated by 5G-MEDIA MANO to implement a generic MEC-MANO hardware and a software platform.

With regards to the 5G-MEDIA use cases, the C-RAN platform will serve as a means of improving the performance of the video services for use cases 2 & 3. When video is cached to the edge cloud close to the BBU, it can provide lower latency effects and increase the throughput of the traffic which is needed for this type of service.

Also, for the operator, it is more convenient to manage, secure and administer the network when all elements, (BBUs) and applications, (videos) are close to the edge cloud. The virtualisation of BBUs is also another advantage for the use cases since it is more easily and faster instantiated together with the Virtual Machines (VMs) hosting the video services.

### 2.2.3. Elastic transcoder and interactive 360° media platform

#### 2.2.3.1. Overview

BitTubes Elastic Transcoder, shown in Figure 11, is a highly scalable dockerized transcoder service with a build in load balancing mechanism. It encodes the most common video files into an adaptive bitrate streaming format based on Apple’s HTTP Live Streaming protocol (HLS). The transcoder uses the Hypertext Transfer Protocol (HTTP/HTTPS) for the communication between the internal components, as well as for external access, via providing a number of appropriate REST API endpoints. This leverages the ease of operation mainly by its uniform
interface, thus ensures a very wide support for any kind of infrastructure to be integrated. The key facts of the transcoder are:

- Adaptive bitrate streaming (MPEG2HLS)
- HTTP/HTTPS transfer
- User-based authentication ( Principals, Roles, ACLs )
- REST API
- Load balancing
- High scalability
- On-premises

Figure 11 – BitTubes elastic transcoder

The Interactive 360° Media Platform is a Software as a Service which enables a novel interactive, clickable 360° video experience for all platforms. The goal is to support the creative industry to engage immersive video and storytelling experience at the viewer. As shown in Figure 12, this is achieved by the fact that the content producer manually enriches new or already existing video content with supplemental information or creates multiple storylines and side-stories by branching enriched video content. It gives the viewer the opportunity to interact with objects that are part of the video via certain user interactions (e.g. click, touch, gaze). Thereby, it leverages the viewer’s personal involvement, while content provider and
broadcaster may benefit from getting anonymous viewer statistics such as click, touch, gaze behaviour, the viewing path or the most frequently visited scenes. Such a combination of interactive 360° video and analytics leads to value-added information for content creators to create content smart and to break new ground of storytelling.

2.2.3.2. Relevance to 5G-MEDIA use cases and platform architecture

With respect to 5G-MEDIA, the transcoder will be considered as one of the media-specific VNFS. In principle, all considered UCs in 5G-MEDIA are connected by a certain functional requirement that simply describes the encoding/decoding of live video streams. This can be achieved by BitTubes Elastic Transcoder service, which can be brought in to 5G-MEDIA's overall platform architecture as media-specific VNF. Note that while it still needs adaption for live streaming and broader codec support in the 5G-MEDIA context, by now it fulfils in its dockerized container format the requirements for Function-as-a-Service (FaaS) paradigm.

2.2.4. Serverless computing

2.2.4.1. Overview

Apache OpenWhisk (incubating) is a cloud native distributed serverless open source platform able to execute application logic (Actions) in response to events (Triggers) from external sources (Feeds) or HTTP requests governed by conditional logic (Rules). It provides a programming environment supported by a REST API-based Command Line Interface (CLI) along with tooling to support packaging and catalogue services. This programming paradigm is schematically represented in Figure 13.
The main entities of OpenWhisk concept are:

- **Event Sources**: devices, queues, databases, HTTP calls, and webhooks emitting classes of events in the form of triggers. In 5G-MEDIA, events can be generated by applications, MANO stack, and different management components of the platform itself.
- **Triggers**: triggers are the class of events (including device readings, published messages, HTTP call parameters, and data changes) that are emitted by event sources.
- **Actions**: actions are functions that encapsulate code. Actions are written in any supported language by implementing a single method signature to be executed in response to a trigger. A special case of action is a black box action, which is a Docker container created from some base image offered by OpenWhisk. In 5G-MEDIA, we provide a specialized base image for the black box actions, so that the non-standard features, such as support for network communication and Day 1 configuration of FaaS VNFs are supported out of the box for the 5G-MEDIA customer (e.g. an application developer).
- **Rules**: rules represent the declarative association between a trigger and an action, defining which action(s) should be executed in response to an event.
- **Sequences**: sequences allow to chain together a sequence of actions in a declarative
fashion with output of one action in a sequence being an input of the next one\(^\text{13}\).

- Packages: packages encapsulate external services in a reusable manner and assemble them into triggers and actions.

A developer writes a function in a high-level language of her choice (currently supported by OpenWhisk are node.js, Python, Java, Scala, Go, Swift, and PHP with more language support on the way). Next, a programmer calls OpenWhisk CLI to create an action. Under the hood, OpenWhisk saves the action metadata and code in a database. Now the action can be invoked either directly or through a rule (which also entails defining a trigger and an event source first). For the sake of simplicity, we assume that a newly created action is invoked directly (i.e., through a REST call to the OpenWhisk API end-point). To understand what happens under the hood, consult the OpenWhisk architecture shown in Figure 14.

\[\text{Figure 14 – OpenWhisk architecture}\]

An action invocation is a REST call to NGINX\(^\text{14}\), a de facto industry standard for reverse proxy and Level 7 load balancer that surfaces an end point to an OpenWhisk service. NGINX terminates SSL and forwards the request to Controller. The latter consults CouchDB about permissions of the user who wishes to invoke the action (shown as the light blue coloured arrow). Each action, rule, trigger, package, sequence, generically referred to as artefact lives

\[^{13}\text{It should be noted that this capability allows to pass small amounts of data from one action to another transparently via a database, but this feature is insufficient for streaming media over the network. Hence, we added network support for the OpenWhisk actions.}\]

\[^{14}\text{https://www.nginx.com/}\]
in a name space associated with the user and for which an API key is allocated for the user to access artefacts in this name space.

If the user’s action invocation request successfully passes permissions checking, Controller selects one of the Invokers, a containerized Scala code running on a Docker box (referred to as invoker machine) and publishes an action invocation request on a Kafka topic to which this Invoker is subscribed. The load balancing mechanism executed by Controller aims at publishing an action invocation request to Invoker that has the highest chance of having a warm container that can execute this action. The Invoker has a fixed (configurable) number of slots for executing action containers (by default, Invoker has 16 slots, which means 16 containers can be executed simultaneously on the invoker node). In case there is no warm container a cold start happens, which adds a latency of container creation to the overall action invocation latency.

When Invoker picks up a request for action execution from the Kafka topic, the serverless “magic” happens. The Invoker fetches the action metadata and code from the CouchDB (remember that the action was created via the OpenWhisk CLI, as a precursor to this scenario) and passes it to Web service running in the container that will execute the action. If such container does not exist, it is created first. The Web service of the container serves two resources: /init and /run. A REST Call from the Invoker to /init causes the Web service of the action container to place the action code to the right place in the file system of the container. This is referred to as dynamic code injection. Next, the Invoker issues a POST HTTP request to the /run resource and the previously injected code is executed. The Invoker monitors execution of the action container. When the execution finishes, the Invoker puts the result in the CouchDB and the Controller is informed that the action has terminated. A user can asynchronously fetch the results from the database using action invocation ID that was produced upon action invocation. In another mode, a user can invoke an action synchronously and the results will be returned by controller upon the action completion. The Controller layer is horizontally scalable and many controllers can be added under NGINX load balancer to avoid single point of failure and performance bottlenecks.

Apache OpenWhisk community includes IBM, RedHat, Adobe and a number of other companies. Being a mature open source platform, OpenWhisk is well suited for innovation in serverless computing, which is also known as Functions-as-a-Service (FaaS). Apache OpenWhisk is production-ready and available as a service via IBM Cloud Functions¹⁵ and Adobe as I/O runtime¹⁶. It is also integrated with Red Hat's OpenShift¹⁷ and deployed on-premises in several organization worldwide. It is possible to provide OpenWhisk

---

¹⁵ https://ibm.biz/openwhisk
¹⁶ https://www.adobe.io/apis/cloudplatform/runtime.html
¹⁷ https://developers.redhat.com/blog/2018/05/16/summit-faaS-openwhisk-openshift/
PaaS powers on top of any IaaS or within a Data Center infrastructure on VMs or bare metal machines. Also, OpenWhisk can be deployed as a group of containers on top of container orchestrator engine, such as Kubernetes (K8s)\(^{18}\). This is a particularly important deployment option, because it makes OpenWhisk deployment cloud-neutral and aligns with the de-facto industry standard for container orchestration. Also, K8s provides support for networking, resource discovery, and scheduling policies out of the box. These capabilities are crucial in enabling the use case scenarios, where actions (implementing different VNFs in a chain) should:

- Discover each other;
- Communicate with each other over the network to stream media;
- Be placed on execution nodes with special capabilities (e.g., having GPUs) to support quality of experience;
- Be placed on the execution nodes under affinity/anti-affinity constraints, to ensure correct operation and QoS for VNFs;
- Be discoverable by external services.

It should be emphasized that these features are not supported in OpenWhisk out of the box and their introduction is an important innovation achieved by 5G-MEDIA. As part of our exploitation plan we plan submitting pull requests for merging these features upstream in Apache OpenWhisk aiming at making them readily available as part of the standard Apache OpenWhisk distribution for the benefit of the industry. One particularly important deployment option is deploying OpenWhisk on top of Kubernetes.

Apache OpenWhisk architecture targets large scale cloud service deployments for FaaS (tens of thousands of concurrent action invocations). However, event-driven programming model being facilitated by OpenWhisk can also be deployed at the edge to save bandwidth, reduce latency by bringing computation closer to the data sources and allow for autonomic operation. Edge environment can be of many different types at one extreme edge can be a fully capable data center, while at another extreme it can be a resource constrained gateway, such as Raspberry Pi. To that end IBM Research - Haifa develops a Lean OpenWhisk and advances it in the Apache OpenWhisk community to be merged upstream\(^{19},^{20}\).

Figure 15 shows the Lean OpenWhisk architecture. The main differences from the standard OpenWhisk architecture are as follows:

- There is no Kafka;
- Controller directly communicates with Invoker using over a function call via in-memory object;
- Controller and Invoker comprise a single entity;

\(^{18}\) https://kubernetes.io/

\(^{19}\) https://medium.com/@davidbr_9022/lean-openwhisk-open-source-faas-for-edge-computing-fb823c6bbb9b

\(^{20}\) http://mail-archives.apache.org/mod_mbox/openwhisk-dev/201807.mbox/browser (search Lean OpenWhisk)
• Service Provider Interface (SPI) is used to dynamically load a LeanLoadBalancer class that we provide (this class facilitates direct communication with Invoker).

These changes result in small memory footprint (a tenfold reduction from that of the regular OpenWhisk) and high performance at the resource constrained edge.

In 5G-MEDIA we use Lean OpenWhisk as part of SDK to allow emulation of VNFs implemented as OpenWhisk actions.

![Diagram of Lean OpenWhisk Architecture](image)

_Figure 15 – Lean OpenWhisk Architecture_

It is important to stress that Lean OpenWhisk is not intended to be a new project. Rather it is a special distribution of Apache OpenWhisk that is built from the current code base. Thus, once Lean OpenWhisk is merged upstream, it will be sustainable and maintained by community seamlessly. We aim at making Lean OpenWhisk available through Apache Open Source Incubator in 2018.

2.2.4.1. Relevance to 5G-MEDIA use cases and architecture

An innovation of 5G-MEDIA project is the support of “on demand VNF instantiation” using serverless computing programming model. OpenWisk meets this demand (typically, in the form of containers), where VNF provisioning and deprovisioning is handled by a FaaS platform transparently guaranteeing seamless elasticity and scalability. The 5G-MEDIA SVP treats FaaS platforms as virtualization platforms that are managed by their corresponding VIMs. To that end, an OW VIM will be implemented and be integrated with 5G-MEDIA MANO offering FaaS as a virtualization option that can be combined with other forms of virtualization. Thus, 5G-MEDIA service developers will be able to mix and match the most suitable technologies to attain cost-efficiency NS development.
2.2.5. Symphony

2.2.5.1. Overview

Symphony is the NXW’s platform for the integration of home/building control functionalities, devices and heterogeneous subsystems. Symphony can monitor, supervise and control many different building systems, devices, controllers and networks available from third-party suppliers. By intelligently correlating cross-system information, a flexible and highly efficient platform is delivered to the stakeholders. The system is a service-oriented middleware integrating several functional subsystems into a unified IP-based platform. As hardware/software compound, Symphony encompasses media archival and distribution, voice/video communications, home/building automation and management, and energy management. The system allows to create scenarios for the control of lighting and curtains; to control the climate in all the rooms; to manage TV and phone calls. The supervision module integrates the management of all the monitoring systems of both internal and external areas, through customizable graphical console.

Figure 16 – The Symphony unified software platform

Symphony can communicate with any automation controls, both standard protocols and proprietary systems. It is a comprehensive cloud-based software suite of building management tools. It allows to monitor and control diverse building automation systems, by integrating different protocols under a coordinated, unified management level with an open and modular approach.
Symphony can monitor, supervise and control many different building systems, devices, controllers and networks available from third-party suppliers, as shown in Figure 18. By intelligently correlating cross-system information, a flexible and highly efficient platform is delivered to the stakeholders. The Symphony management station in the cloud allows operations, administration and management of the Building Management Systems (BMS) from any authorized remote terminal. The innovative BMS-as-a-service paradigm provides a scalable service architecture, data security and privacy, customized dashboards and business intelligence. To guarantee maximum confidentiality, BMS can be deployed on a private cloud infrastructure.

As shown in Figure 19, the Symphony Video on Demand component allows to offer to your guests thousands of video and music contents (including audio/video indexes and metadata) through the widest choice of A/V devices, all seamlessly integrated with any display, TV screen, amplifier and speaker currently on the market, as well as with all the major remote controls.
(like Crestron, AMX) and with iPhone, iPad devices.

![Symphony media component functionalities](image)

**Figure 19 – Symphony media component functionalities**

### 2.2.5.2. Relevance to 5G-MEDIA use cases and architecture

The Symphony VoD component will be a relevant key feature in the context of the 5G-MEDIA use cases. In particular, for Use Case 3 “Ultra-high Definition over Content Distribution Networks”, the Symphony media server, capable of archiving, managing and distributing different type of media content, such as video, music and images with different encoding, will be virtualised and packaged as a virtual application to be deployed in an automated manner through the 5G-MEDIA SVP. In the CDN service workflow, the virtualised Symphony media sever will take the role of the origin server, enabling the distribution of UHD video on demand through the virtual caches hierarchy in the CDN domain. Finally, the packaged Symphony media server application will be available in the 5G-MEDIA NSD/VNFD catalogue, allowing developers to include this component in their novel media services to be validated, deployed and operated through the 5G-MEDIA SVP.

### 2.3. Summary of exploitable tools and solutions

Table 1 summarizes the exploitation and the integration plan of the 5G-MEDIA project for the most suitable achievements and outcomes from 5G-PPP phase 1 projects.
Table 1 – Exploitation and integration plan of 5G-PPP phase 1 outcomes in 5G-MEDIA

<table>
<thead>
<tr>
<th>Tool</th>
<th>Use in the 5G-MEDIA architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>COGNET</td>
<td>Exploitation of MAPE common infrastructure, dataflow and deployed technology to realize 5G-MEDIA cognitive control and develop services of CNO. Extensions related to AI algorithms, modernization of tools and advanced functionality.</td>
</tr>
<tr>
<td>SELFNET</td>
<td>Exploitation of technology used for catalogues and repositories to store VNFs and NSs, organize metadata and provide ETSI-compliant VNF/NS onboarding process. Extensions towards multi-MANO support (with a focus on ETSI-compliance with OSM) and integration with 5G-MEDIA SDK.</td>
</tr>
<tr>
<td>SONATA</td>
<td>Exploitation of SDK tools for VNFs/NSs editing, validation and emulation. Emulation of FaaS and extension of descriptors and records with respect to monitoring and different packaging and technologies</td>
</tr>
</tbody>
</table>

Table 2 summarizes the exploitation and the integration plan of the most suitable technologies, frameworks and commercial solutions, from those offered by the 5G-MEDIA partners, in the scope of 5G-MEDIA project.

Table 2 – Exploitation and integration plan of commercial solutions in 5G-MEDIA

<table>
<thead>
<tr>
<th>Tool</th>
<th>Partner</th>
<th>Use in the 5G-MEDIA architecture</th>
<th>Integration plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnLife platform</td>
<td>TID</td>
<td>Provide an OSM integrated NFVI environment</td>
<td>Integration of OpenNebula VIM with 5G-MEDIA SVP and integration of monitoring mechanisms to MAPE loop of 5G-MEDIA</td>
</tr>
<tr>
<td>C-RAN</td>
<td>OTE</td>
<td>Provide C-RAN resources as a NFVI. C-RAN follows a centralized architecture that facilitates network management through NFV.</td>
<td>Implement a generic MEC-MANO hardware and software platform that could serve as a means of improving the performance of the networkservices related to 5G-MEDIA use cases.</td>
</tr>
<tr>
<td>Elastic transcoder</td>
<td>BIT</td>
<td>Enable highly scalable dockerized transcoder service</td>
<td>Implement as media-specific VNF (e.g. vTranscoder) used in 5G-MEDIA Use Cases.</td>
</tr>
<tr>
<td>Interactive 360° Media Platform</td>
<td>BIT</td>
<td>Enable interactive, clickable 360° video experience</td>
<td>Implement as media-specific VNF (Relates to the CognitiveServices referred to Use Case 2 and maybe Use Case 3)</td>
</tr>
<tr>
<td>Tool</td>
<td>Partner</td>
<td>Use in the 5G-MEDIA architecture</td>
<td>Integration plan</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OpenWhisk</td>
<td>IBM</td>
<td>Enable serverless computing</td>
<td>Integration of OW with 5G-MEDIA SVP (in the form of VIM) and SDK (extend SDK emulator).</td>
</tr>
<tr>
<td>SYMPHONY</td>
<td>NXW</td>
<td>Provide Media server implementation for Use Case 3. By exploiting symphony’s services for media archival and distribution, voice/video communications, home/building automation and management, and energy management</td>
<td>Implement Symphony media streaming for Use Case 3 as VNFs</td>
</tr>
<tr>
<td>Apache Traffic Control</td>
<td>NXW</td>
<td>Integrate open source open CDN platform composed of caching elements and CDN monitoring tools to develop VNFs for edge transcoding units and edge caches and provide CDN monitoring &amp; traffic stats services.</td>
<td>Implement as media-related VNF</td>
</tr>
</tbody>
</table>
3. Evaluation of candidate open source solutions for 5G-MEDIA platform

Two core components of the 5G-MEDIA architecture are the Application Service Development Kit (SDK) used for development and verification of NSs/VNFs, and the Management and Orchestration (MANO) framework used to enable the NFV/SDN paradigm. Since for both these needs there are several open source solutions as well outcomes of 5G-PPP phase 1 projects, we briefly analyze the advantages/disadvantages of the most promising and well-known candidates in this section and evaluate their appropriateness in the scope of the 5G-MEDIA project.

3.1. Evaluation of service development kit tools

In general, an SDK is a set of tools that allows the creation of applications and supports developers in implementing, packaging, and deploying the software. In close analogy to software development kit, we use the term SDK for the tools that is going to be used, developed or enhanced in the scope of 5G-MEDIA project to provide media related service development.

The 5G-MEDIA Application SDK aims at providing a set of tools that helps media application developers to easily implement and deploy new network applications to the SVP. The 5G-MEDIA SDK comprises:

- Editors for generating media related network apps.
- Model checkers and validators.
- Support of packaging tools and easy catalogue access.
- Deployment to SVP platforms.
- Emulators for executing trials.
- Serverless toolkit to enable the flexibility to seamlessly augment a chain of pre-packaged VNFs with custom functions developed in arbitrary programming languages. In addition, to provide additional value to developers, serverless computing paradigm will be analysed and integrated to SDK tools.

In order to meet all these features, we will reuse existing open source software and tools as much as possible. In the following, we present the main specifications of three of the most recognized development frameworks for 5G NFV platforms, i.e. i) the SONATA SDK, ii) the Open-Baton User Tools, and iii) the OSM Design Time Tools.

3.1.1. SDK candidate solutions

In 5G-MEDIA, the SDK will be designed taking into account the development of media related applications. In general, it will serve to the different roles of the developer ecosystem including network and media function developer, service developer and end-user application developer. In order to simplify the development process and provide powerful tools to the developer, the 5G-MEDIA SDK will comprise of editing, validating, monitoring and profiling capabilities. Network Function Virtualization Orchestrator (NFVO) exposes counterparts of these functionalities, in an ETSI NFV compatible NFV platform. In such platform, more specifically, on-boarding, lifecycle, performance, fault, and package management of a network service is done by the NFVO through the Os-Ma-Nfvo reference point by employing restful
interfaces. Therefore, in this section in order to reveal the usefulness of already existing solutions in application development for ETSI NFV compatible platforms we will consider the northbound of the network virtualization platforms focusing the solutions near the surrounding of NFVO and its Os-Ma-Nfvo reference point. Three candidate solutions for 5G-MEDIA SDK are considered, i.e. SONATA SDK, Open-Baton User Tools and OSM Design Time Tools.

3.1.1.1. SONATA SDK

SONATA SDK contains a set of loosely coupled lightweight tools and a sequence or combination of execution of these tools assist the developer in designing network services. These tools mainly targeted to work together with the SONATA Service Platform (SP) but they are encouraged to be used by other projects as well.

Os-Ma-Nfvo interface in ETSI reference architecture is mapped to SDK-Gk (interface between SDK and Gatekeeper) and Bss-Gk (interface between BSS and Gatekeeper) interfaces in SONATA\textsuperscript{21}. The main interface between the SDK and the Sonata Platform, namely SDK-Gk is over the Gatekeeper. Gatekeeper controls which users or systems can access the features provided by the SONATA Platform (such has onboarding or downloading packages, services or functions meta-data, downloading records, subscribing for (a)synchronous monitoring, etc.), onboarding packages (after successful validation) into the Catalogue, accepting service instantiation requests, etc.\textsuperscript{22}.

The main components of the SONATA SDK are organized into the following two toolkits based on their role, functionality and provided services:

1. Programming tools
   - Schema (son-schema)
   - Editors (son-editor)
   - Validators (son-validate)

2. Packaging and integration tools
   - Descriptors Packaging (son-package)
   - Access tools (son-access)
   - Emulator tools (son-emu)
   - Monitoring (son-monitor)
   - Profiling tools (son-profile)
   - Analyzing Tool (son-analyze)

The role and the provided services of each tool of the above toolkits are briefly presented in the following subsections.

\textsuperscript{21} D2.3 Updated Requirements and Architecture Design, Sonata Project, December 8th, 2016
\textsuperscript{22} D5.4 Final release of SONATA platform, Sonata Project, 04/08/2017.
3.1.1.1.1. Programming tools

3.1.1.1.1.1. Schema tool (son-schema)
Son-schema contains the schema files for the various descriptors used by SONATA as well as some examples and tests. The schema files in this repository serve as ground truth for the whole SONATA project. The schema tool supports the various descriptors (VNFD/NSD) used by the SONATA service platform, which are compliant to the ETSI NFV specifications.

3.1.1.1.1.2. Editor tool (son-editor)
The SONATA Editor (son-editor) is a web-based application whose frontend is running in the network service developer’s browser. It is compatible with the SONATA eco-system and makes use of tools from the SONATA SDK. Its main purpose is to assist in the creation and editing of SONATA network service projects and their descriptors. This is done by simplifying repetitive and complicated tasks e.g. the descriptors of NSs are visualized as a graph of VNFs.

son-editor provides the following services to NS developers:

- Assists in the creation and editing of application/service/function descriptors.
- Visualizes VNFs as a graph by showing nodes and connections between them.
- Allows users to analyse nodes and its descriptors in a graphical layout.
- Handles workspace (son-workspace), project creation(son-project) through CLI tools.
- Integrates with GitHub and allows developers to share developed application/service/function projects or public repositories.

son-editor user interface supports following functionalities:

- Provides following graphical network service editing capabilities:
  - Drag and Drop network components.
  - Connect by dragging connections.
  - Undo and redo, zoom and pan, multi-select.
  - Automatic computation of the forwarding graph (optional).
- Provides following immediate syntax validation capabilities:
  - Visual feedback what needs to be filled and what is still missing.
  - Easy to select constrained values if specified by schema.
- Allows checking following reference dependencies of VNFs:
  - Hinder deletion of referenced VNFs.
  - Create new version or refactor references when renaming VNFs.
- Allows uploading packages directly to SONATA-SVP in following ways:
  - Package via son-package tool.
  - Upload services directly from network service view.
  - GitHub integration.
  - GitHub OAuth login.
  - Clone, share, pull and push projects from and to GitHub.
3.1.1.1.3. **Validator tool (son-validate) and features**

The SONATA validator is the tool responsible for validation of SDK projects, packages, services and functions. In addition, the SONATA SP is using the son-validate service API for the validation of packages. The son-validate also features a GUI tool which is embedded in son-editor enabling an easier way of validating and visualizing services.

*son-validate* provides the following services to NS developers:

- Enables the customization of validation issues to be reported by a user-defined level of importance such as error, warning or none.
- Validates SDK projects, packages, services and functions in terms of structure, syntax, integrity and network. For instance, the service descriptors are validated against schema templates or the network topology is validated in terms of connectivity logic such as unlinked VNFs, Virtual Development Units (VDUs) and connection points, network loops/cycles and node bottlenecks.
- Validation UI has the following capabilities:
  - Validation triggering

![Figure 20 – Validation selection UI](image-url)
• Issue source highlight

![Figure 21 – Forwarding graph issue highlight](image)

• Cycle identification and highlighting.

![Figure 22 – Forwarding graph cycle identification](image)

3.1.1.1.2. Packaging and integration tools

3.1.1.2.1. Descriptors packaging (son-package)

The *son-package* tool has the main role of packaging a project, making it ready and available for instantiation in the SONATA SP. To be able to solve external dependencies, *son-package* interacts with son-access which is responsible for retrieving service and function descriptors from the SP’s Catalogues. It is integrated with *son-validate* to perform the validation process, not only on the syntax, but also on the integrity and topology. As a result, during the generation of a package, *son-package* validates all the required components to make sure the package is valid. Package information such as vendor, name, version, maintainer and description can be configured in the SDK project configuration. A package does not necessarily require having a service ready to be instantiated, it can also serve as a vehicle to share and publish descriptors for later collaboration.
3.1.1.1.2.2. Access tool (son-access)

The son-access component provides a secured connection based on authentication and authorization processes between SDK end-users and the SONATA SP, which offers possibilities to use end-user credentials and JSON Web Tokens to access the SONATA Service Platform features such the unified SP Catalogue and enable end-users to submit and request package files and descriptors from the SP Catalogue.

3.1.1.1.2.3. Emulator tool (son-emu)

son-emu supports network service developers to locally prototype and test complete network service chains in realistic end-to-end multi-PoP scenarios. It allows the execution of real network functions, packaged as Docker containers, in emulated network topologies running locally on the network service developer’s machine. Son-emu provides following functionalities to network service developers:

- Emulation capabilities to the NFV developer such as modifying and evaluating the placement process, and/or assessing potential service scaling alternatives.
- Supports SONATA’s own Service Platform and the MANO solution provided by the OSM project.
- Supports monitoring and profiling components of SONATA SDK, enabling to test both functional as well as performance-related characteristics of services.

3.1.1.1.2.4. Monitoring tool (son-monitor)

son-monitor guides the network service developer by providing performance data of the service and its components in a quantitative manner. son-monitor supports following features:

- **Metrics Gateway and Websocket interface**: This is a mechanism to stream data from the SONATA Service Platform to the SDK.
- **External Service Access Points**: The son-emu supports External Access Points that can be used to attach traffic generating processes or scripts that running outside of the emulation environment, on the host itself or on a different machine.
- **VNF Resource Starvation Monitor**: This implements a detection mechanism to timely stop the profile run when the host machine gets overloaded.

3.1.1.1.2.5. Profiling tool (son-profile)

son-profile allows developers to deploy network services on SONATA’s emulation platform and do some load testing under different resource constraints. During these tests a variety of metrics can be monitored which allows service developers to find bugs, investigate problems or detect bottlenecks in their services. The main purpose of son-profile is to automate big parts of this workflow to support network service developers as much as possible.
3.1.1.1.2.6. Analyzing tool (son-analyze)

son-analyze provides statistical tools such as R statistical environment or Python Jupyter scientific libraries to analyze profiling and monitored service data. This tool is also integrated with son-access to authenticate to SONATA’s SP.

3.1.1.2. Open-Baton SDK

Open-Baton\textsuperscript{23} is an open source platform providing an implementation of the ETSI NFV MANO specification. The main interface of the Open-Baton is its NFVO REST API, which is made available in Java through a class called Open-Baton SDK. As mentioned above, 5G-MEDIA SDK is a comprehensive combination of tools, which is beyond being solely an interface to NFVO. Similarly, as an alternative, Open-Baton also provides a CLI that can be used to invoke messages to NFVO REST API. On the other hand, the dashboard of Open-Baton gives more control options to the users about the lifecycle of different objects like VIM Instances, Network Service Descriptors/Records and Virtual Network Function Descriptors/Records. However, these options are not specifically designed for the developers’ perspective. They provide monitoring and status information through the NFVO.

3.1.1.3. Open Source MANO design time tools

In Open Source MANO (OSM)\textsuperscript{24} Release THREE, the exposed API and endpoints on the top of the NFVO are categorised in two groups: The first group assembles a user interface VNF Package Generator and a VNF/NS Catalogue Composer. The second group assembles the components related with the DevOps, the major of which are the Installer, the Jenkins and the Emulator.

3.1.1.3.1. Design time components in user interface

The VNF Package Generator is a tool that assists VNF providers to create a properly formed package for on-boarding in OSM.

The VNF/NS Catalogue Composer is a model-driven Graphical User Interface that supports both VNF providers and Operator’s Network Service designers to rapidly develop descriptors that accurately represent the essence of the entity being modelled for deployments.

3.1.1.3.2. DevOps components

In order to deliver an excellent experience for OSM developers, OSM Release THREE has taken a significant step in terms of DevOps components with the creation of a new module responsible for the Continuous Integration and Continuous Development (CI/CD) workflow\textsuperscript{25}.

---

\textsuperscript{23} https://openbaton.github.io/

\textsuperscript{24} https://osm.etsi.org/

\textsuperscript{25} https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseTHREE-FINAL.PDF
OSM CI/CD pipeline which is given in Figure 23 has been built to provide support for community collaboration, packaging in Docker containers and testing operations within Docker containers, smoke testing and also end-to-end system tests leveraging real NFV infrastructure, VIMs, SDN controllers and VNFs. As in Figure 23, OSM CI/CD pipeline has been built with four stages which are:

- **Stage 1** focuses on Gerrit which supports community collaboration, i.e. code reviews. The first stage in the OSM CI/CD pipeline allows for a single Gerrit trigger (i.e. a code commit) to initiate a multi-branch pipeline defined in Stage 2.

- **Stage 2** involves a per-module pipeline where the testing work related to a specific module exists. The second stage operates tests within Docker containers and allows for parallel execution of Gerrit pipelines. (All of the OSM modules now support packaging in Docker containers.) Module Development Groups (MDGs) only need to implement their respective call-backs for test, build and archive functionality. This stage also drives the automated code license scan and module specific unit tests, builds packages and archives the created artifacts. JFrog Artifactory was leveraged to manage artifacts outside of the master Jenkins node.

- **Stage 3** focuses on system installation and smoke testing. The installation is based on the binary build artifacts created during stage 2. Fresh OSM installs are created inside an LXC container for executing smoke tests. “Smoke” testing is performed as a simple and rapid set of end-to-end system-oriented tests such as API checks and VNFD

---


27 https://jfrog.com/artifactory/
uploads that verify basic OSM functionality continues to work. Smoke testing does not depend on NFVI, VIMs or SDN Controllers.

- Stage 4 incorporates end-to-end system tests leveraging real NFV infrastructure, VIMs, SDN controllers and VNFs. It works in conjunction with ETSI’s Hub for Interoperability and Validation (HIVE) testing environment to enable testing on geo-distributed, multi-VIM environments.

Running extensive orchestration tests may cause hysteresis effects on the VIM and NFVI environment. To support the development of the CI/CD pipeline for extensive MANO test runs, an emulator providing multiple PoP NFVI environments has been added to the DevOps repo.

### 3.1.2. Evaluation criteria, assessment and reasoning

In this section we evaluate the previously presented SDKs. The purpose of this evaluation process is to highlight the advantages/disadvantages of the different candidates, as well their tools that can be re-used and/or enhanced in the scope of 5G-MEDIA. After presenting the used evaluation criteria, we assess each candidate SDK and make the final decision about which is the best option to become the basis of 5G-MEDIA SDK design.

#### 3.1.2.1. Evaluation criteria

The criteria used to evaluate the SDK’s described above are the following:

- **Technical documentation and support for installation and maintenance:** Typically, contributors in open source projects lean towards to be more involved in coding than documenting. Documentation as either internal (i.e. comments in source code) or external (i.e. library documentation, user documentation, etc.) can greatly affect the re-usability of the software. Adequate documentation allows others to understand and modify the software which generates the supporting community.

- **Community Support:** Active open source projects usually have a community with common interests in continuously evolving a related product or in using the results of the open source project. Therefore, community support generally comes with the existence of related active products/projects or useful and timely results of the open-source project that rises attention of contributors. SourceForge\(^\text{28}\) categorizes stage of product development into six: Planning, Pre-Alpha, Alpha, Beta, Production/Stable and Mature. Except the planning phase, these development stages are passed not only by development but also by testing. Field tests and user community can be considered as the means for the maturity of the software.

\(^{28}\) Krishnamurthy, S. (2002). Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. First Monday, 7
• **Open Source**: In general, open source software satisfies three fundamental criteria\(^{29}\): i) free redistribution, ii) availability of source code, and iii) right to create derived works. Every open source software is available under a certain type of license that specifies the terms and conditions under which the source code can be used, modified and/or shared.

• **5G MEDIA SDK-DevOps Requirements Compliance – Packaging Tools**: In the sense of packaging, 5G-MEDIA SDK will be equipped with tools that turn the code in the source repository to a running application or service that can be deployable to a testing or product environment. ETSI recently started to define and specify a common VNF package format, based on the TOSCA Cloud Service Archive (CSAR) standard\(^{30}\). Being ready to support TOSCA CSAR will have positive impact on the evaluation. From the continuous integration standpoint, having DevOps as an integrated part of the project, 5G-MEDIA SDK aims to provide tools that developer can benefit from to deliver network applications, functions and services. For this purpose, a repository that will allow multiple developers to collaborate easily will be available. For each commit to this repository, static code analysis quality (i.e. proofing) supporting different programming languages will be satisfied and schema definitions of the service descriptors will be verified. User interfaces of these functionalities will be an integrated part of the editor in 5G-MEDIA SDK and for the sake of modularity validator will be a separated identity. Therefore, schema validators and syntax checkers that can be reused or adapted will be very valuable for 5G-MEDIA.

• **5G MEDIA SDK-DevOps Requirements Compliance – Monitoring Tools**: Both in production site and in the local emulation environment to observe the KPIs and track the issues, it is necessary to have means of monitoring. Monitoring will be an integrated part of the SVP, but the 5G-MEDIA SDK will provide a user interface for service monitoring therefore we also considered the design time components of evaluated projects for the monitoring aspects of 5G-MEDIA SDK.

• **5G MEDIA SDK-DevOps Requirements Compliance – Profiling Tools**: Before the 5G-MEDIA application developer deploys the application/service, the profiling is used to verify functionality or performance of developed components in the local emulation environment under one or more multiple resource constraints. Then, this functionality is also evaluated as a selection criterion for the alternative SDKs.

• **5G MEDIA SDK-DevOps Requirements Compliance – Emulator Tools**: Since the availability and consistency of the production site is critical, an emulation platform is very beneficial to locally test the functionality in continuous integration phase before the real deployment. Therefore, support of an emulation environment in the considered projects is evaluated as well.

---

\(^{29}\) Online: [https://opensource.org/docs/definition.html](https://opensource.org/docs/definition.html) Accessed on 06.01.2018.

\(^{30}\) Thomas Spetzier Gerd Breiter, Frank Leymann. TOSCA: Cloud Service Archive (CSAR). Website, July 2012.
• **5G MEDIA SDK-DevOps Requirements Compliance – FaaS**: FaaS, also called serverless computing, is an emerging paradigm enabling you to run code without the need to provision a machine or a container. Just provide your code, upload it to an end-point, define triggers and let the platform handle the rest (from scaling to failures). Therefore, FaaS tools are also evaluated for the alternative SDKs.

• **Interoperability with the MANO**: Declares the interoperability of the alternative MANOs in the 5G-MEDIA project, by providing complete services for VNFs/NSs development and deployment.

### 3.1.2.2. Assessment of candidate solutions

Table 3 summarizes the assessment of the two considered SDKs with respect to the evaluation criteria set in the previous subsection.

**Table 3 – Assessment of SONATA, Open-Baton and OSM SDKs**

<table>
<thead>
<tr>
<th>No</th>
<th>Criteria</th>
<th>SONATA SDK</th>
<th>Open-Baton SDK</th>
<th>OSM SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Technical documentation and support for installation and maintenance</td>
<td>Installation Documentation(^{31}) Administration Documentation(^{32}) Tutorials available(^{33})</td>
<td>Documentation(^{34})</td>
<td>Documentation(^{35})</td>
</tr>
</tbody>
</table>

\(^{31}\) [https://sonata-nfv.github.io/component_installation](https://sonata-nfv.github.io/component_installation)

\(^{32}\) [https://sonata-nfv.github.io/start_using](https://sonata-nfv.github.io/start_using)

\(^{33}\) [https://github.com/sonata-nfv/son-tutorials](https://github.com/sonata-nfv/son-tutorials)

\(^{34}\) [http://openbaton.github.io/documentation/](http://openbaton.github.io/documentation/)

\(^{35}\) [https://osm.etsi.org/wikipub/index.php/Main_Page](https://osm.etsi.org/wikipub/index.php/Main_Page)
<table>
<thead>
<tr>
<th>No</th>
<th>Criteria</th>
<th>SONATA SDK</th>
<th>Open-Baton SDK</th>
<th>OSM SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Community Support</td>
<td>Key partners of SONATA are part of 5G PPP phase 2, 5GTANGO project that extends SONATA technical outcomes. This will also improve the maturity of implemented Agile DevOPS Model.</td>
<td>Open-Baton is mainly progressed with research projects by Fokus and TUBS and has a small developer community.</td>
<td>OSM SDK sustainability is high because the project is hosted by ETSI. Developers and users’ community is very large under the flagship of ETSI OSM OSG.</td>
</tr>
<tr>
<td>3</td>
<td>OpenSource</td>
<td>Yes (Apache 2.0 License)³⁶</td>
<td>Yes (Apache 2.0 License)³⁷</td>
<td>Yes (Apache 2.0 License)</td>
</tr>
<tr>
<td>6</td>
<td>5G Media SDK-DevOps Requirements Compliance – Packaging Tools:</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>5G Media SDK-DevOps Requirements Compliance – Monitoring Tools</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>5G Media SDK-DevOps Requirements Compliance – Profiling Tools</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

³⁶ Source codes available at: [https://github.com/sonata-nfv](https://github.com/sonata-nfv)
³⁷ Source codes available at: [https://github.com/openbaton/openbaton-client](https://github.com/openbaton/openbaton-client)
### Table 3

<table>
<thead>
<tr>
<th>No</th>
<th>Criteria</th>
<th>SONATA SDK</th>
<th>Open-Baton SDK</th>
<th>OSM SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>5G Media SDK-DevOps Requirements Compliance – Emulator Tools</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>5G Media SDK-DevOps Requirements Compliance – FAAS</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>Interoperability with OSM v3.0</td>
<td>Custom integration is needed.</td>
<td>Custom integration is needed.</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Interoperability with OPNFV</td>
<td>Custom integration is needed.</td>
<td>Custom integration is needed.</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>Interoperability with Open-Baton</td>
<td>Custom integration is needed.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>Interoperability with SONATA v3</td>
<td>Yes</td>
<td>Custom integration is needed.</td>
<td>Custom integration is needed.</td>
</tr>
</tbody>
</table>

### 3.1.2.3. Selection and reasoning

In the scope of service development for the 5G-MEDIA Project, reusability of components in three different projects are evaluated. As given in Table 3, these are SONATA SDK, Open-Baton User Tools and OSM Design Time Tools. 5G-MEDIA SDK is envisaged as a comprehensive combination of tools (packaging, profiling, emulator, monitoring, FaaS etc.), which interacts with the NFVO through a gatekeeper. From this perspective if we look at the alternatives:

Open-Baton user tools do not give sufficient developer tools; user tools in Open-Baton are designed based on the REST API of NFVO, for instance the SDK in Open-Baton is a library in Java which exposes the REST API and CLI is yet again uses the REST API in the background. Dashboard as a part of user tools in Open-Baton provide monitoring and status information but it is not specifically designed for the developers.

On the other hand, OSM design time components provide tools (VNF Package Generator and VNF/NS Catalogue Composer) related with packaging and catalogue management. These tools can be re-used especially with taking advantage of their compatibility since 5G-MEDIA is going to integrate the OSM platform. The DevOps approach in OSM is for the purpose of its own development but it can be taken as a reference in 5G-MEDIA for the network service development capabilities that will be given to the external developers.
SONATA SDK is a combination of several tools especially designed for the service development in 5G. However, interoperability with the OSM, which is considered as a candidate to be used as the MANO in the 5G-MEDIA (Section 3.2.2.3), is not supported by SONATA-SDK out-of-the-box except the emulator tool. This tool was adopted by OSM project as part of their DevOps MDG under its new name *vim-emu*. Nevertheless, OSM Design Time Tools user interfaces manage packaging and on-boarding of VNFs and NSs to the OSM. From this point of view, some of the software components of OSM Design Time Tools might be either re-used or adapted as the packaging tool inside 5G-MEDIA SDK. Consequently, the SONATA SDK enriched with the OSM Design Time Tools will be the basis of the SDK in 5G-MEDIA project.

### 3.2. Evaluation of MANO frameworks

In this section, the following four open source frameworks are evaluated:

- Open-Baton,
- Open Source MANO (OSM),
- Open Platform for NFV (OPNFV), and
- SONATA,

as candidate solutions for 5G-MEDIA MANO. For each considered framework, we present its architectural approach and main specifications and then we assess its candidacy by using a specific list of critical evaluation criteria.

#### 3.2.1. MANO Candidate solutions

##### 3.2.1.1. Open-Baton

Open-Baton is an open source project by Fraunhofer FOKUS and Technical University of Berlin which implements a comprehensive ETSI NFV MANO layer compliant with the ETSI MANO functional specification\(^3\).

---

\(^3\) [https://openbaton.github.io/](https://openbaton.github.io/)
Open-Baton provides many features and components briefly summarized in the following list:

- A NFVO completely designed and implemented following the ETSI MANO specification.
- A generic Virtual Network Function Manager (VNFM) and a generic EMS able to manage the lifecycle of VNFs based on their descriptors.
- A Juju VNFM Adapter in order to deploy Juju Charms or Open-Baton VNF Packages using the Juju VNFM.
- A driver mechanism supporting different type of VIMs without having to re-write anything in the orchestration logic.
- A powerful event engine based on a pub/sub mechanism for the dispatching of the lifecycle events execution.
- An autoscaling engine which can be used for automatic runtime management of the scaling operations of the VNFs.
- A fault management system which can be used for automatic runtime management of faults which may occur at any level.
- A network slicing engine which can be used to ensure a specific QoS for Network Services.
- A monitoring plugin integrating Zabbix as monitoring system.
- A Marketplace useful for downloading VNFs compatible with the Open-Baton NFVO and VNFM.
- A set of libraries (in Java, Go and Python) which could be used for building a derived specialized VNFM.

A more detailed representation of the Open-Baton architecture is provided in Figure 25.
The NFVO is the main component in Open-Baton, implemented in Java and spring.io framework. Currently Open-Baton provides a VIM plugin compatible with OpenStack allowing its seamless integration with the NFVO. The NFVO supports the addition of more than one VIM based on the same or different implementations. Open-Baton’s architecture encourages its extension, and alternative VIMs can be supported via a plugin mechanism and a dedicated SDK library for implementation of VIM-specific drivers to be easily integrated with the NFVO. Open-Baton provides a generic VNFM as well as a set of additional SDK libraries to be used for the development of a specific VNFM. The generic VNFM implementation follows the ETSI MANO specification, using an EMS to interact with a given VNF. The communication between the VNFM and the NFVO or EMS is handled through a message bus (RabbitMQ). If the current VNFM implementation does not address a specific use case scenario, a new VNFM may be developed using the SDKs provided by Open-Baton. The way that this VNFM interacts with each VNF has no restrictions from Open-Baton so the workflow and communication format is free to be defined by developers. The interaction between the developed VNFM and the NFVO is performed via a REST interface or AMQP protocol message system. Open-Baton follows the ETSI MANO specifications for VNFs and uses template descriptors for the definition of the resources (compute, storage, and network) as well as the apps (VNFs, NSs) to be managed, allowing the exchange and persistence of this information.
Open-Baton’s modular implementation and its plugin-based architecture allows extensibility and the integration of the solution without having to modify or understand its core implementation.

3.2.1.2. Open source MANO

ETSI OSM is an operator-led ETSI community that is delivering a production-quality open source MANO stack aligned with ETSI NFV Information Models and that meets the requirements of production NFV networks.

The OSM community has set itself the goal of being a world-class production ready solution. OSM Release THREE\(^\text{39}\) represents another significant step along this path. It has been engineered, tested and documented to be functionally complete to support Operator RFx processes, and to be a key component for internal/lab and external/field trials as well as interoperability and scalability tests for virtual network functions and services. It allows for rapid installation in VNF vendor, system integrator and operator environments worldwide. OSM Release THREE substantially enhances interoperability with other components (VNFs, VIMs, monitoring tools, SDN controllers) and provides a plug-in framework to make platform maintenance and extensions significantly easier to provide and support.

Building on the capabilities developed for prior releases, Release THREE improves administrator and developer experience, both in terms of usability and installation procedure as well as the modelling of VNFs and network services. In line with the goals of the OSM project, the output of this modelling work has been contributed to ETSI NFV. Release THREE also provides extremely flexible VNF configuration and advanced networking management as well as improved security capabilities, with an advanced access control system.

3.2.1.2.1. OSM scope

The OSM community has defined an expansive scope for the project covering both design time and run-time aspects related to service delivery for telecommunications service provider environments. The express goal is that the OSM code base can be leveraged in these environments as-is in a Roll-Your-Own context, or in whole and/or part of a commercial product offering.

Figure 26 shows the approximate mapping of scope between the OSM components and the ETSI NFV MANO logical view (the background image was extracted from Figure 4 in the NFV Reference Architecture Framework, ETSI GS NFV 002 V1.2.1 (2014-12)).

\(^{39}\) https://osm.etsi.org/images/OSM-Whitewpaper-TechContent-ReleaseTHREE-FINAL.PDF
The runtime scope of OSM includes:

- An automated Service Orchestration environment that enables and simplifies the operational considerations of the various lifecycle phases involved in running a complex service based on NFV.
- A superset of ETSI NFV MANO where the salient additional area of scope includes Service Orchestration but also explicitly includes provision for SDN control.
- Delivery of a plugin model for integrating multiple SDN controllers.
- Delivery of a plugin model for integrating multiple VIMs, including public cloud based VIMs.
- Delivery of a plugin model for integrating multiple monitoring tools into the environment.
- One reference VIM that has been optimized for Enhanced Platform Awareness (EPA) to enable high performance VNF deployments.
- An integrated “Generic” VNFM with support for integrating “Specific” VNFM.
- Support to integrate Physical Network Functions into an automated Network Service Deployment.
- Being suitable for both Greenfield and Brownfield deployment scenarios.
- GUI, CLI, Python based client library and REST interfaces to enable access to all features.
The design-time scope of OSM includes:

- Delivery of a capability for Create Read Update Delete (CRUD) operations on the Network Service Definition.
- Supporting a Model-Driven environment with Data Models aligned with ETSI NFV MANO.
- Simplifying VNF Package Generation.

Supplying a Graphical User Interface (GUI) to accelerate the network service design time phase, VNF on-boarding and deployment.

3.2.1.2.2. OSM release THREE

OSM Release TWO has made significant steps in advancing on each of the themes noted above. However, there is one theme in particular that is worthy of further mention in the context of a Release THREE overview, i.e. the production readiness.

The very explicit goal of the community is to enable production-ready deployments in operator networks. OSM Release THREE represents a significant milestone along this path. The bar the community is setting for itself before applying a “deployment ready” label is high. Nonetheless, the leadership team feels comfortable in advising operators that they should now consider this release to be of sufficient capability as a framework for PoCs and trials. Operators can also use OSM Release THREE to progress their RFx processes.

The full list of distinct capabilities that have been progressed can be found on the OSM website and WIKIs. The following section notes the salient categories of innovation that articulate the cohesion for this release.

- **Security:** One of the challenges of deploying an NFV based network service is to make sure the MANO environment is compatible with the organizational structures that exist in the operator environment. It is required that the visibility into the network and the control of network operations presented to different actors employed by the operators can be suitably configured to match their roles in the organization. This separation of visibility/control of the network is required for secure and robust operation of the network. To address these requirements, OSM Release THREE has added comprehensive Role-Based Access Control and Multi-Tenancy/Project to the interface model.

- **Service Assurance:** An assured network service delivery environment requires the ability for running network services to be able to scale-out their support level as well as the ability to scale-in if the need for the additional capacity is no longer required. OSM Release THREE includes support for network scaling events to add and remove full VNFs from a running Network Service.

- **Resiliency:** A resilient service delivery platform is required for production readiness. OSM Release THREE has made considerable progress on this topic by improving the recovery on single component failure, supporting multiple VCA instances and by offering improved scalability of the OSM platform.

- **Usability:** OSM continues to focus on being an easy to use MANO platform. The Python-based OSM client offers a straightforward method to interact with the most
commonly used OSM operations. Most VNF consoles are now accessible via the GUI. And, once installed, the user can update to maintenance releases without re-installation.

- **Interoperability**: One of the guiding principles of OSM is that each component is both replaceable and pluggable. To that end, Release THREE has taken another substantial step forward to drive interoperability with other components (VNFs, VIMs, SDN controllers, monitoring tools). At the VIM level, an Amazon Web Services EC2 plugin was extended, further improvements were added to the VMware vCloud Director VIM plugin compatible with vCloud NFV 1.5 and 2.0, and support for VMware Integrated OpenStack (VIO) was delivered. At the SDN controller level, the SDN controller support received further incremental refining for ONOS as well as the OpenDaylight (ODL) and Floodlight. For service assurance, the new monitoring plugin framework enabled Amazon CloudWatch, VMware vRealize Operations Manager, OpenStack Aodh and OpenStack Gnocchi monitoring tools.

### 3.2.1.3. Open platform for NFV

Open Platform for NFV (OPNFV) delivers an open-source networking stack to accelerate the deployment of NFV components and network services in enterprise service and telecom providers. OPNFV facilitates the development, integration and testing of different NFV components combinations and versatile application-driven scenarios (from mixed web, network and IT points of view) across various open source ecosystems and flavors of technology stacks. Its key contribution includes:

- The provision of a reference NFV platform for telecom and enterprise networks in the paradigm of a Systems Integration Platform. The platform is designed, established and evolved as an open community effort. Currently, OPNFV combines OpenStack and over 30 additional selectable components and creates a master repository for 45+ (upstream) projects, where each one aims to serve different needs and requirements of NFV.

- The delivery of platform that meets the operational needs of industry and enterprise businesses. Especially that considerably limits accelerating time to market for NFV solutions and enables stakeholders’ choices in specific technology components based on their architectural preferences, used technologies and deployed scenarios.

- The establishment of an end-to-end stack to support several NFV scenarios with verified capabilities and characteristics, which establishes agile reference methodologies (requirements, documentation and propagation; continuous integration, testing, and continuous delivery), and offers a process and supporting tools for testing and validating NFVI and MANO products and solutions.

OPNFV organizes its NFV stack business over the three pillars shown in Figure 27[^40].

The activities within each one of the three pillars shown in the above figure are:

**Integration**: OPNFV integrates a variety of open source projects (upstream projects) to address specific NFV requirements. When the community contributes software code, this is usually integrated directly to the relevant upstream project. Figure below shows the relationship between OPNFV projects and the many upstream projects collaborated with. As shown in Figure 28, OPNFV integrates multiple projects for the same purpose. In this way, it enables alternative options for the end users to deploy scenarios of their own interest.

**Testing**: OPNFV tests the entire NFV stack across a variety of NFV-specific parameters. Each individual (feature) project must manage its own test strategy including unit, functional, security, and performance tests, while it is also responsible of the associated troubleshooting. Nevertheless, several specific test projects validated by OPNFV already deal with defining specific testcases, perform tests not covered by a single project, create tooling and study end-to-end performance. Figure below shows the universal testing ecosystem established by
OPNFV and the related test specific projects, which aim to help OPNFV projects to be integrated in CI to meet test criteria for the release\(^{41}\).  

**New features:** For each upstream open source project, OPNFV serves as the vehicle for NFV requirements. By actively working upstream and providing a single voice for NFV requirements, OPNFV steers these open source projects to serve the needs of NFV. Test projects, shown in Figure 29, are in charge to run their own test cases and help feature projects to be integrated in CI to meet test criteria for the release.

![Figure 29 – Test projects in OPNFV](image)

### 3.2.1.3.1. OPNFV platform

OPNFV introduces a comprehensive platform for policy-driven orchestration and automation of network services and functions providing the following NFV qualities\(^{42}\):

- A common mechanism for life-cycle management of VNFs, which include deployment, instantiation, configuration, start and stop, upgrade/downgrade and final decommissioning.
- A consistent mechanism for specifying and interconnecting VNFs, VNF chains and PNFs in an agnostic way of the physical network infrastructure, network overlays, etc., i.e., virtual link.
- A common mechanism for dynamically instantiating new VNF instances or decommissioning sufficient VNF instances to meet the current performance, scale and network bandwidth needs.
- A mechanism for detecting faults and failures in the NFVI, VIM and other components of the infrastructure and recovering from those failures.

\(^{41}\) [https://wiki.opnfv.org/display/testing/TestPerf](https://wiki.opnfv.org/display/testing/TestPerf)

\(^{42}\) [https://www.opnfv.org/software/technical-overview](https://www.opnfv.org/software/technical-overview)
● A mechanism for sourcing/sinking traffic from/to a physical network function to/from a virtual network function.

● NFVI as a Service for hosting different VNF instances from different vendors on the same infrastructure

OPNFV initially focused on building NFVI and VIM by integrating components from upstream projects, such as OpenDaylight, OpenStack, Ceph Storage, KVM, Open vSwitch, and Linux (releases Arno/Brahmaputra). More recently, OPNFV has extended its portfolio of forwarding solutions to include fd.io and ODP, is able to run on both Intel and ARM commercial and white-box hardware, support VM, Container and BareMetal workloads, and includes MANO components primarily for application composition and management (release Danube). As upper MANO stack components are further integrated, it is expected that the diversity of those components will blur boundaries/roles of the ETSI-referenced elements e.g. NFVO, VNFM, EMS, OSS/BSS. In Figure 30, the proposed enhanced reference platform for MANO integration in OPNF is shown43. OPNFV has introduced a special Work Group to coordinate integration of MANO into OPNFV and effort of different upstream projects.

---

43 https://wiki.opnfv.org/display/mano/MANO+Group+Home
It should be noted that OPNFV does not recommend or integrate any specialized SDK solution upon the MANO component. Thus, services such as NFV related accounting, NS/VNF lifecycle management requests, data analytics exchange and other OSS/BSS actions are addressed per se by each MANO component used in the deployed scenarios. Also, that functional testing of VNFs, VNF integration and VNF deployment is out of OPNFV scope at this time. However, some (functional) tests (e.g. introduced by upstream project Functest) take advantage of open source VNFs to generate real-life test conditions on underlying VIM and NFVI components. A list of such VNFs can be found here.\footnote{https://wiki.opnfv.org/display/functest/List+Of+VNFs}

\footnote{https://wiki.opnfv.org/display/functest/List+Of+VNFs}
3.2.1.3.2. Danube release

On average, a major OPNFV platform release is performed per six-month period, each named like a river. The key features of the most recent release Danube (after Arno, Brahmaputra and Colorado releases), include the following aspects:\(^{45}\):

- **Foundational support and introduction of capabilities for MANO**: Integration between NFV Infrastructure/Virtual Infrastructure Manager (NFVI/VIM) with Open-Orchestration (Open-O) platform (now ONAP) through Opera upstream project and Open-Baton platform through Orchestra upstream project; instrumentation of NFVI network telemetry to support Service Assurance and other use cases; multi-domain template support (Domino project); and translation features between YANG and Tosca modeling languages (Parser project). OPNFV is also interested in working with OSM as well, however this is not currently supported by the Danube release.

- **Enhanced DevOps automation and testing methodologies**: Danube brings a fully integrated CI/CD pipeline, the creation of Lab-as-a-Service (LaaS) (leveraged by Pharos community Labs project) to enable dynamic provisioning of lab resources, the introduction of stress testing into the OPNFV test suite, and a Common Dashboard that provides a consistent view of the testing ecosystem.

- **Focus on NFV performance**: Acceleration of the data plane via FD.io integration (a high-performance alternative to OVS) for all Layer 2 and Layer 3 forwarding (FastDataStacks project), and continued enhancements to OVS-DPDK and KVM. The release Danube also sees a renewed focus on performance test project activities through virtual switch testing (VSPERF project), root cause analysis for platform performance issues (Bottlenecks projects), initial compute subsystem performance testing to lay the groundwork for Benchmarking as a Service (QTIP project), and storage subsystem performance testing (Storperf project).

- **Key NFV architectural enhancements**: Danube launches the ability to dynamically enable and configure network control through integration with OpenStack Gluon and increased reliability and test cases that support multi-site and High Availability (HA) work.

- **Feature enrichment and hardening in core NFVI/VIM functionality**: Typical such functionalities are the IPv6, Service Function Chaining (SFC), L2 and L3 Virtual Private Network (VPN), fault management and analysis, and a continued commitment to support multiple hardware architectures, as well as traditional hardware OEMs, whitebox, and open source hardware through collaboration with the Open Compute Project.

- **Experimental support for a VIM based on Kubernetes**: Kubernetes has been introduced in Danube. The open source container orchestration project OPNFV OpenRetriever aims to improve native support of Kubernetes integration and testing for the next release (Euphrates).

---

\(^{45}\) [https://www.opnfv.org/software](https://www.opnfv.org/software)
3.2.1.4. SONATA

3.2.1.4.1. SONATA service virtualization platform

The SVP introduced by SONATA project is shown in Figure 31\(^{46}\). Its general architecture design complies with ETSI reference architecture for NFV management and orchestration. In SONATA SVP, lifecycle management operations are divided into service-level and function-level operations. These two domains define the elements that build the NFVO and the VNFM functionalities in ETSI’s reference architecture. The key reference points of ETSI NFV are preserved (e.g., the Or-Vnfm interface) and complemented (e.g., Vi-Vnfm interface responsible for WAN infrastructure management) in SONATA.

As shown in Figure 31, SONATA SVP is composed of four main components; 1) the gatekeeper which is the main entry point of the SVP. The gatekeeper implements API endpoints for SDK tools, like the packaging tool, and allows service developers to manually control and manage services deployed on the platform, 2) a platform-specific catalogue which stores service artefacts uploaded to the platform, 3) the repositories used to store meta data of running services, e.g., monitoring data or placement results, and 4) SONATA's extensible MANO framework which interfaces with other components and implements all management and orchestration functionalities of the service platform.

---

\(^{46}\) SONATA Consortium. Sonata deliverable 2.2: Architecture design.
3.2.1.4.2. SONATA MANO framework

The core of SONATA’s SVP is a highly flexible MANO framework that provides all functionalities to manage complex network services throughout their entire lifecycle. This framework uses a message broker system as flexible communication backend to establish communication and message exchange among the different artefact and software components. It provides functionalities like message translation between sender and receiver, authentication, or different messaging patterns. Upon the brokering mechanism, the framework implements service management, as well as VNF management, using a set of loosely coupled functional blocks, which are called MANO plugins. This design allows platform operators to easily integrate new functionalities into the platform by adding new MANO plugins to it. The behaviour of some of these plugins can be customized by service developers, with so-called service or function specific managers, which are bundled with a service package and uploaded to the platform. This allows service developers, for example, to define service specific scaling and placement algorithms.

Figure 32 shows the MANO framework of SONATA consisting of a set of loosely coupled components, called MANO plugins. Each of these plugins implements a limited, well-defined part of the overall management and orchestration functionality. This design has the advantage that functional parts of the system can easily be replaced, which provides a high degree of flexibility. With this, for instance, the platform operator is able to customize the platform’s functionality or to add new features by adding additional plugins.
3.2.1.4.2.1. Topic-based communication

As mentioned above, the MANO framework’s internal communication is based on a message broker and uses a topic-based publish/subscribe pattern. This enables each component to talk to all other components without the need to configure or announce API endpoints among them. All components that want to connect to the system have to register themselves to the message system, based on RabbitMQ system, that controls which messages are allowed to be sent and received by the component. This is achieved by a topic-based permission system defining read/write permissions for each connected component. This makes the implementation asynchronous and, as mentioned above, the integration of new orchestration plugins is much simpler than it is in architectures with classical plugin APIs. The interactions that require a request/reply-based communication behaviour, e.g., allocation requests of compute resources, are also done on top of the used publish/subscribe pattern by implementing two distinct topics to which requests and the corresponding replies are published. SONATA doesn’t enforce a specific programming language for implementing plugins, neither prescribes how the plugins are executed. For example, plugins may be executed as operating system-level processes, containers, like Docker, or separate virtual machines within the platform operator’s domain.

3.2.1.4.2.1. Plugin management and coordination

MANO plugins are the main components of the MANO framework and are under the control of the platform operator. They can be executables, containers, or VMs which typically run inside the domain of the platform operator and connect to the main message broker. The only requirement a MANO plugin has to fulfil is the ability to communicate with the used messaging system.

To keep track of the connected MANO plugins, the system uses a plugin management component, which can either be an extension of the used message broker system or an external component that interacts with the configuration interface of the message broker.
This component is responsible for managing service platform-related components and implements the following functionalities:

- Authentication: it authenticates plugins that want to connect to the message broker system.
- Plugin registration: It registers authenticated plugins, so the system becomes aware of their presence. A plugin also announces its capabilities and functionalities to the plugin manager during the registration procedure. It announces, for example, that it implements the lifecycle management logic for a certain type of VNF.
- Topic permission management: The plugin manager is also responsible to control the topic permissions that specify which plugins are allowed to publish or subscribe to a certain topic.
- Bookkeeping: The plugin manager maintains a dictionary of active plugins and all functionalities registered by them. Each plugin can query this dictionary to obtain information about available functionalities in the system. A service manifest manager can, for example, query for a plugin that is able to parse a specific service description format.
- Watchdog functionality: All plugins are executed as independent entities that can fail. The plugin manager is responsible to monitor the system and check for failed plugins that do not respond anymore. If it detects a failed plugin, it can generate events to inform other components about the issue and it can restart the failed plugin.

3.2.1.4.2.2. Function and service specific management and orchestration

A key feature of the MANO framework is the ability to customize the management and orchestration behaviour for single services and network functions. This is realized with SSMs and FSMs. Such managers are small management programs implemented by a service/function developer with the help of SONATA’s SDK. They have a similar lifecycle like MANO plugins, including separated connection and registration procedures. The difference to MANO plugins is that FSMs/SSMs are provided by function/service developers and not by the platform operator. This makes their management and coordination more complicated. Typical examples for such specific managers are custom service scaling algorithms or function lifecycle managers.

SSMs and FSMs are shipped together with a network service or function inside the package uploaded to the service platform. Just like any other artefact contained in a service package, they are statically checked by the gatekeeper module and extracted from the package before they are forwarded to other components of the platform, e.g., stored in catalogues. They connect to specialized MANO plugins, which provide an interface to customize their default behaviour. Such plugins are called executive plugins. It should be noted that the platform operator always keeps control over its own platform behaviour, even if parts of its functionality are customized by SSMs/FSMs implemented by the service developers.

Executive plugins provide a clear separation between FSMs/SSMs and the rest of the management and orchestration system. An advantage of this is that each executive has fine-grained control over messages sent to and received from an FSM/SSM. A placement executive
can, for example, filter and change topology information used as input to customized placement SSMs. The reason for this can be that the platform operator would not want to share the original network topology with the platform customers. Furthermore, executive plugins are responsible to check all outputs of FSMs/SSMs to detect misbehaviour, e.g., a scaling SSM may not be allowed to request scale-out operations with hundreds of new VNF instances. In such cases, outputs of customized SSMs are discarded by the executive and the results of the default FSM/SSM can be used. The interface between executive plugins and FSM/SSM is defined by the executive plugin developer and tailored to the tasks it should provide. Only the FSM/SSM boarding and management procedures are pre-specified as shown in the next section. Note that each executive plugin in SONATA’s service platform can be customized by multiple FSMs/SSMs. Each of these FSMs/SSMs belongs to exactly one platform customer and manages one or multiple functions/services of that customer. This design makes it necessary to isolate the communication between FSMs/SSMs in order to ensure that the managers of one customer cannot influence or monitor the managers of other customers.

3.2.2. Evaluation criteria, assessment and reasoning

3.2.2.1. Evaluation criteria

The criteria used to evaluate the MANO frameworks described above are the following:

- **Sustainability**: Sustainability describes MANO’s potential to have a significant impact in the current and future 5G NFV/SDN landscape. A key aspect to evaluate sustainability is the communities that have already committed to support further development, maintenance, and production of the considered MANOs.

- **Supported NFVI environment**: Lists the virtualization and cloud computing frameworks that are compatible with the MANO framework (i.e. the necessary VIM exists to control/manage the corresponding virtualized infrastructure).

- **Technical documentation and support for installation and maintenance**: The set of available documents and leaflets that help developers to install, execute and develop code.

- **Resources required for installation**: Describes the infrastructure resources required for the proper installation and deployment of the MANO framework (e.g. in terms of CPU, memory, disk, necessary software packages etc.).

- **Software requirements**: Describes the software resources and IT technologies required for the proper installation and deployment of the MANO framework.

- **Integration with SDK tools**: Declares the availability of advanced SDK tools providing complete services for VNFs/NSs development and deployment, which are fully compatible with the considered MANO.

- **Project lifecycle and governance**: Describes the authorities that undertake MANO governance (i.e. the strategic task of setting its goals, direction, priorities, limitations etc.) and lifecycle management (i.e. allocation of resources, communication, day-to-day operations etc.).

- **List of already available VNFs**: Includes the list of VNFs that are provided as open source and can be reused or extended by 5G-MEDIA.
● **Support VNF chaining**: Declares whether MANO supports configurable and flexible SFC, VNF orchestration and traffic routing.

● **Support VNFFGs update during runtime**: Declares whether MANO supports online updates of already deployed VNFFGs due to changes in the network conditions and demands, e.g. need for dynamic horizontal scaling, fast state migration etc.

● **Support hypervisor- and container-based technologies**: Lists the hypervisor-based and container-based technologies that are supported by the MANO.

● **Background expertise**: Declares whether previous experience exist among the partners of the 5G-MEDIA project.

### 3.2.2.2 Assessment of candidate solutions

Table 4 summarizes the performance of the four considered MANO frameworks wrt to the criteria analysed in the previous subsection.

**Table 4 – Assessment of candidate solutions for 5G-MEDIA MANO**

<table>
<thead>
<tr>
<th>No</th>
<th>Criteria</th>
<th>OSM v3.0</th>
<th>OPNFV</th>
<th>Open-Baton</th>
<th>SONATA rel. v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sustainability</td>
<td>Yes, developers’ and users’ community is large and under ETSI OSM OSG flagship</td>
<td>Yes, it is a Linux Foundation project with support from all big firms</td>
<td>Fraunhofer Fokus and TUBS are supporting. Developers’ community is small</td>
<td>Yes. Supported by 5G PPP partners involved in 5GTANGO.</td>
</tr>
<tr>
<td>2</td>
<td>Support NFVI environment</td>
<td>OpenVIM OpenStack VMware Amazon Web Services</td>
<td>OpenStack</td>
<td>OpenStack Test VIM driver</td>
<td>OpenStack</td>
</tr>
<tr>
<td>3</td>
<td>Technical documentation and support for installation and maintenance</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Criteria</td>
<td>OSM v3.0</td>
<td>OPNFV</td>
<td>Open-Baton</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Resources required for installation</td>
<td>Memory 16GB (bare metal deployment): Jumphost, controller, compute. For Jumphost: Memory 16 GB CPU cores 32, Hard Disk 500 GB</td>
<td>3 servers: Jumphost, controller, compute. For Jumphost: Memory 16 GB CPU cores 32, Hard Disk 80 GB</td>
<td>Minimal version: &gt; 2GB of RAM, and &gt; 2 CPUs, 10GB of disk space Complete version: &gt; 8GB of RAM, and &gt; 8 CPUs, 10GB of disk space</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Software requirements</td>
<td>Ubuntu 16.04 Desktop LXD containers</td>
<td>Ubuntu 14.04 or Ubuntu 16.04 or Debian Jessy</td>
<td>Ansible 2.3.0+ Git Docker CE 17.06</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Integration with SDK tools</td>
<td>No</td>
<td>No</td>
<td>Yes (there is an SDK for implementing specific VNFMs)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Project lifecycle and governance</td>
<td>OpenSource project ETSI member Governance is managed via several groups.</td>
<td>Linux foundation</td>
<td>OpenSource project on GitHub, but roadmap fully in FOKUS &amp; TUBS hands</td>
<td>5G PPP member. SONATA open source is supported by 5GTANGO.</td>
</tr>
</tbody>
</table>
### No | Criteria | OSM v3.0 | OPNFV | Open-Baton | SONATA rel. v3
---|---|---|---|---|---
8 | List of already available VNFs | No | VNFs out of scope (for testing purposes) | Some Open source VNFs in OB marketplace⁴⁷ | A few VNFs related to the use cases, as described in the project deliverables.
9 | Support VNF chaining | Yes | Yes | Yes | Yes
10 | Support VNFFGs update during runtime | Yes | No | No | Yes, but with predefined templates.
11 | Support hypervisor- and container-based technologies | LXD | Yes (KVM, LXD) | Yes | Yes. However, containers are used for SONATA Service Platform components, while VNFs are initialized within regular OpenStack VMs.
12 | Background expertise | Yes | No | Yes | Yes

### 3.2.2.3. Selection and reasoning

#### 3.2.2.3.1. Sustainability

**OSM**: OSM sustainability is high because the project is hosted by ETSI. Developers and users community is very large under the flagship of ETSI OSM OSG. Currently, OSM has 40 member companies and 42 participant companies, which are shown in the ETSI OSM organization web⁴⁸.

---


⁴⁸ [https://portal.etsi.org/TBSiteMap/OSM/ListofOSMMembers.aspx](https://portal.etsi.org/TBSiteMap/OSM/ListofOSMMembers.aspx)
**OPNFV:** OPNFV sustainability is highly guaranteed since the project is a core member of the open source networking ecosystem governed by Linux foundation, including OpenDaylight, OVS, ONAP, FD.io etc. Currently, OPNFV membership includes 49 member-companies (including AT&T, CISCO, ERICSSON, HP, NOKIA, IBM etc.). Also, the project is strongly influenced and interacts with many upstream projects supported by several open source and specialized communities. Note, finally, that the Linux Foundation and the Technical Steering Committee of OPNFV constantly provide tools, training and events (e.g. open source networking days) to scale up community interest and disseminate its evolution and new achievements to the widest audience.

**Open-Baton:** Open-Baton sustainability is mainly related to the continued support by Fraunhofer Fokus and TUBS The community of developers is small and currently it is not yet defined if and how contributions from external parties (other than the Open-Baton development team at Fokus and TUBS) could be integrated. Open-Baton is mainly progressed with research projects by Fokus and TUBS.

**SONATA:** It is true that within the SONATA ecosystem, sustainability cannot be considered as an a priori relaxed condition. However, there are two reasons that seem like assuring sustainability; the first one is related to the support in terms of sustainability and visibility offered by multiple service and network operators, vendors and large companies across Europe (such as Telefonica, British Telecom, NEC, NOKIA, Atos, THALES), surrounded by SMEs, as part of the consortium. Second, sustainability is addressed by the fact that the key partners of SONATA are part of 5G-PPP phase 2, 5GTANGO project that extends SONATA technical outcomes.

### 3.2.2.3.2. Support of NFVI environment

**OSM:** OSM supports, in release THREE, four options as VIM: OpenVIM⁴⁹, Openstack⁵⁰, VMware vCloud Director⁵¹ and Amazon Web Services AWS⁵². OSM can also manage external SDN controllers to perform the dataplane underlay network connectivity on behalf of the VIM⁵³.

**OPNFV:** The OPNFV integrates Openstack as VIM. In addition to OpenStack, the Danube release has experimental support for a VIM based on Kubernetes, which enables containerized VNFs to be used in NFV deployments. The use of Kubernetes in the Danube release does not include any SDN controller integration, nor is it integrated with any other NFV-centric project. It is expected Kubernetes integration to deepen over subsequent releases.

---


**Open-Baton:** Open-Baton supports Openstack as VIM\(^{54}\). Also, a test VIM\(^{55}\) is supported only for testing purposes as a VIM mockup. It is possible to build a VIM driver, using the plugin-sdk provided by Open-Baton\(^{56}\).

**SONATA:** According to the information provided by the project website, in its latest release 3.0\(^{57}\), SONATA supports OpenStack for instantiating network services and hosting VNFs, followed by their respective Service Function Chain (also known as VNF Forwarding Graph). Moreover, each of the individual MANO framework components is installed on Docker containers\(^{58}\). This means that Docker containers technology is supported by SONATA, although an effort during the project lifetime to integrate the support for Kubernetes was only partially successful, due to problems that are mentioned in\(^{59}\), such as security and networking issues. Nonetheless, the SONATA project, as a proof of concept, has implemented deployment of its network functions on Docker using its emulation platform.

Complementary to the supported VIM/NFVI environments, in SONATA, the management and orchestration of the deployed services and functions is facilitated through the use of plugins, i.e. SSMs/FSMs, which is one of the novelties of the project. Service specific managers are responsible for scaling and managing whole services, whereas the function specific managers operate at the level of individual functions. SSMs and FSMs are part of the NFVO and the VNFM, respectively.

3.2.2.3.3. Technical documentation and support for installation and maintenance

**OSM:** All documentation about release THREE of OSM is available at the Wiki of the organization\(^{60}\). On this site, a complete explanation of the installation and configuration process is presented, with the help of some technical videos explaining the steps to proceed. The guide includes an example to run the first VNF, however, for the moment we couldn’t set it up.

**OPNFV:** OPNV has launched the project “Documentation” which is responsible to create and update documentation for each platform release, develop guidelines and tooling for documentation across all OPNFV projects, and also maintain consistent documentation libraries. The introduced documentation policy focuses especially on the following aspects:

- All documentation should be available for review early enough in the release process to allow a quality review process.

---

\(^{54}\)[http://openbaton.github.io/documentation/openstack-driver]

\(^{55}\)[http://openbaton.github.io/documentation/test-driver/]

\(^{56}\)[http://openbaton.github.io/documentation/vim-driver-create/]

\(^{57}\)[http://www.sonata-nfv.eu/content/releases]

\(^{58}\)[https://github.com/sonata-nfv/son-mano-framework]

\(^{59}\)[http://www.sonata-nfv.eu/content/pursuit-virtualised-infrastructure-management-based-containers]

\(^{60}\)[https://osm.etsi.org/wikipub/index.php/Main_Page]
• Documents should be tested, meaning assessed per technical correctness, clarity/usability, and presentation quality/consistency.

• OPNFV should strive for continuous improvement of its documentation program, and consistency with best practices for documentation in major open source projects.

It should be noted that apart from official technical documentation and maintenance support, OPNFV provides several tools and updated services to help developers, communities and members of upstream projects to communicate and solve appearing technical issues e.g. wiki pages, website, helpdesk services etc.

The last OPNFV release (Danube) can be installed making use of any of the installation projects in OPNFV: Apex, Compass4Nfv, Fuel or JOID. Each installation project provides detailed step-by-step instructions for working with an installation toolchain and deploying specific scenarios. In principle, an OPNFV installation requires either a physical infrastructure environment, as defined in the Pharos specification and therein provided installation guide, or a virtual one. It is highlighted, though, that installation guidelines are mainly targeting bare metal servers and the process to be followed is quite complicated. During the evaluation period, 5G-MEDIA consortium members tried to install an OPNFV instance and they faced several difficulties and complexities.

**Open-Baton:** All the documentation is available at Open-Baton’s website\(^61\). Following the step by step procedure, it is possible to setup an instance of Open-Baton (as NFVO) and a Generic VNFM. A user mailing list is also available for supporting installation and maintenance.

**SONATA:** It must be highlighted that the documentation provided in order to install SONATA SDK and Service Platform is quite easy to follow\(^62\). By following this guide, we were able to set up the SONATA environment, including both SDK on a laptop and an instance of the Service Platform. However, the guide included some examples that we were not able to execute in the given time frame.

### 3.2.2.3.4. Resources required for installation

**OSM:** OSM needs to be installed Ubuntu 16.04 as base image configured to run LXD containers and at least one network interface. In terms of CPUs, memory and disk the requirements are:

- **Installation from binaries:** 4 CPUs, 8 GB RAM, 40 GB of disk space.
- **Installation from source:** 8 CPUs, 16 GB RAM, 100 GB of disk space.

**OPNFV:** As mentioned above, OPNFV introduces four installation projects, i.e. Apex, Compass4nfv, Fuel and JOID. All of them support installation either on bare metal servers or on a virtual environment, e.g. by using the triple-O tool chain. As a rule of thumb, if only 1 bare metal server is available, then virtual deployment is recommended. If three or more servers are available, then the bare metal deployment is recommended, in which one server is dedicated to the JumpServer (Jumphost), one for the controller, and one for compute

---

\(^61\) [http://openbaton.github.io/documentation/](http://openbaton.github.io/documentation/)

\(^62\) [https://sonata-nfv.github.io/](https://sonata-nfv.github.io/)
processing. In any case, the required resources vary depending on the selected installation project and the targeted scenario, and they may be quite demanding per case. Indicatively, the minimum setup requirements for installing just the Jumphost server by using compass4nfv are 32 CPU cores, 16 or 64 GB of RAM for a bare metal and virtual deployment, respectively, and 500 GB or 1 TB disk for a bare metal and virtual deployment, respectively.

**Open-Baton:** Open-Baton can be installed on any kind of environment (physical hosts, virtual machines, containers, etc.). Suggested requirements in terms of CPUs, Memory, and disk space are:

- minimal version: > 2GB of RAM, and > 2CPUs, 10GB of disk space
- complete version: > 8GB of RAM, and > 8CPUs, 10GB of disk space

**SONATA:** In terms of available resources for installation purposes, SONATA does not need large pool of resources. Although this criterion is not the most important one, since installation will be done once and follow the 5G-MEDIA project throughout its whole lifecycle, it can be considered as a means to evaluate the effectiveness of source code.

### 3.2.2.3.5. Software requirements

**OSM:** No special requirements apart from Ubuntu 16.04 and LXD containers configured. Depending on the VIM chosen, other requirements could be raised, e.g. OpenStack, OpenVIM, ONOS, FloodLight etc.

**OPNFV:** OPNFV platform installation doesn’t require any special software requirement apart from the installation ISO that contains all the necessary packages, e.g. OpenStack, OpenDaylight, ONOS etc. Prior or during the installation process, a number of deployment specific parameters must be configured, e.g. providing sub-net and gateway information, DNS addresses, VLAN and switching information etc.

**Open-Baton:** In case of Linux installation, Ubuntu LTE (14.04 or 16.04) or Debian Jessy is required. The bootstrap script provided with the software automatically downloads, installs and configures all the required software packages (java, rabbitmq, mysql).

**SONATA:** There are no special requirements in terms of software modules. In the github of SONATA, the following software modules are mentioned as pre-requisites: Ansible, Git, Docker, which is a clear advantage compared to other frameworks.

### 3.2.2.3.6. Integration with SDK tools

**OSM:** OSM only integrates a GUI from which VNFs and NSs could be imported and managed. It doesn’t integrate any other dedicated SDK tool for development. However, a web site is provided to generate VNFs packages.

**OPNFV:** OPNFV does not integrate dedicated SDK tools for VNFs/NSs development, packaging, validation, emulation, onboarding or any other related business. All (or part) of these aspects are handled in a different way per use case, depending on the integrated MANO framework

---

63 [http://riftio.com/osm-vnf-package-generator/]
(e.g. ONAP, Open-Baton or Openstack Tacker). Especially for VNF onboarding, OPNFV recognizes the importance of standardizing this process and, thus, enforces the MANO working group along with the project Models to work on issues including VNF package development, VNF package import, VNF validation/testing (basic and in-service), VNF import into a catalogue, service blueprint creation, and VNFD models.

**Open-Baton:** The SDK in Open-Baton consists on the following modules:

- **CLI:** this provides a command-line interface, which enables the use of NFVO’s API and sends commands to it.
- **Open-Baton SDK:** this is the core set of primitives to access the NFVO in a Java application.
- **Plugin SDK:** this is the implementation of the southbound plugin towards VIMs (e.g.: VIM driver for OpenStack)
- **VNFM SDK:** this is the programming framework for writing VNF Managers to be added in Open-Baton.

**SONATA:** Another unique advantage of SONATA is the support of a dedicated SDK that comes directly “out-of-the-box”. The advantage is related to the fact that once a VNF or a NS is validated on the SDK, onboarding and instantiating the service on the SONATA SP is straightforward, including zero-time and no further intervention from the developer. SONATA SDK consists of the following tools: The SDK consists of the following main modules:

- **CLI:** SONATA SDK’s command line interface tools to aid in developing network services and VNFs.
- **Editor:** SONATA’s web-based editor for service and function descriptors. The Editor frontend and Editor backend are the frontend and backend of SONATA’s web-based service and function descriptor editor.
- **Emulator:** emulation platform to support network service developers in locally prototyping and testing complete network service chains in realistic end-to-end multi-PoP scenarios.
- **Analyzer:** analysis framework to study a service’s behaviour.

In the latest release of SONATA open source code, the following features have been added to the SDK: 1) Integrated authentication support in son-access, 2) Graphical User Interface (son-editor) for the creation of network services and associated service and network function descriptors, 3) Inclusion of advanced service validation functionality including graphical debugging tool (son-validate), 4) the emulator (son-emu) has added a prototype implementation of OpenStack Neutron’s SFC chaining API to the emulator. It is of high importance to mention that the emulation platform of SONATA was adopted by ETSI's OSM project as part of their DevOps MDG under its new name vim-emu⁶⁴.

⁶⁴ [https://github.com/sonata-nfv/son-emu](https://github.com/sonata-nfv/son-emu)
3.2.2.3.7. Project lifecycle and governance

**OSM**: The OSM governance structure is made up of a Leadership Group, an End User Advisory Group, a Technical Steering Committee, Module Development Groups and various Task Forces:

- **Leadership Group**: The role of the OSM Leadership Group is to set the policies of the project, to make administrative decisions and to guarantee vendor-neutrality. The members of the Leadership Group coordinate marketing efforts together with the Marketing Task Force and act as the main external representatives for OSM.
- **Technical Steering Committee**: The role of the Technical Steering Committee (TSC) is to coordinate the project’s technical activities.
- **End User Advisory Group**: The End User Advisory Group (EUAG) produces end user recommendations, in the form of feature requests and use cases, to the TSC.
- **Module Development Groups**: An MDG is in charge of developing and maintaining a specific software module/library in the project. It is a technical group, consisting of code contributors (committers) and, among them, a technical leader (MDG Lead) elected by the TSC.
- **Task Forces**: OSM Task Forces are similar in nature to an ETSI Working Group. Task Forces can be created for varying durations depending on the goals, work scope and contributor engagement.

**OPNFV**: The technical work inside OPNFV is done through community-driven projects. Partners are welcome to join existing projects and to propose new ones falling into the categories of integration, testing, and new features. Before their launch, new projects are firstly reviewed and they approved by the Technical Steering Committee. Once approved, they can participate in the OPNFV release cycle under the constraint that they fulfil specific release requirements. This aspect might negatively affect the working plan and deliverables milestones of 5G-MEDIA.

**Open-Baton**: Open Baton is a project developed by Fraunhofer FOKUS and TU Berlin. It is supported by different European publicly funded projects, e.g. NUBOMEDIA, Mobile Cloud Networking, and SoftFIRE. Open Baton represents also one of the main components of the 5G Berlin initiative. Fraunhofer Fokus and Tubs govern the whole roadmap for the project. The software is at its 4th release, published as open source on Github\(^65\). Software maintenance is currently centrally managed by the Open-Baton team at Fokus and TUBS. There is no public mechanism and procedure to include developments by other parties.

**SONATA**: Reference to any particular governance model is observed neither on the SONATA official website, nor on any other relevant document of the project. Compared to other frameworks that define said governance schemes, absence of such plan in SONATA can be seen as an advantage, in the sense that any proposition from the 5G-MEDIA consortium, either as an extension to current open source implementation or continuous support of already

\(^65\) [https://github.com/openbaton/](https://github.com/openbaton/)
implemented features must be decided by the technical committee of the community, leaving space for rejection of the request. Moreover, even in case that a request becomes accepted, it must follow the source code lifecycle, obeying to the specific rules of the open source community. On the contrary, the collaboration with partners involved in SONATA (and most probably also in 5G TANGO) will offer an opportunity to unite and collaborate within the 5G PPP that will offer more freedom on the governance model to be decided.

3.2.2.3.8. List of available VNFs

**OSM:** OSM doesn’t offer a list of VNFs except for an example VNF called CirrOS. However, any VNF structure agreed with ETSI VNF structure is importable to OSM.

**OPNFV:** Functional testing of VNFs, VNF integration or VNF development is out of OPNFV scope. Nevertheless, some tests do take advantage of open source VNFs, such as the vIMS Clearwater, VFW, vAAA etc. or proprietary VNFs which are free for the OPNFV project to use, to build test cases and verify OPNFV platform functionality that covers the VIM and NFVI components (e.g. the project Functest). Additionally, the project Samplevnf aims to use sample open source VNFs for benchmarking and performance optimization.

**Open-Baton:** Open-Baton offers a list of VNFs available at its marketplace. The full list of the VNFs can be found [here](http://marketplace.openbaton.org). In the marketplace there are also a list of NSDs, Vim Drivers (openstack & test) and images. The current list of VNFs contains only free open source VNFs, none of which directly applicable in Media & Entertainment use cases.

**SONATA:** According to Deliverable 6.2[^67], SONATA is using a number of VNFs to support the 3 foreseen pilots, namely: virtual Content Delivery Network (vCDN), Personal Security Application (PSA) and Hierarchical Service Providers (HSP). In the deliverable, the following VNFs are mentioned as part of the pilots: virtual Traffic Classifier, virtual Content Cache, virtual Transcoding Unit, Proxy, Firewall, Intrusion Detection System, virtual Private Network Server, Anonymizer, Traffic Splitter and Merger. It has been identified that some of these VNFs can be used in the 5G-MEDIA Use Cases, accompanied by VNFs already developed by the project partners.

3.2.2.3.9. Support VNF chaining

**OSM:** OSM can manage the lifecycle of VNFs and their chaining in a Network Service according ETSI MANO specifications. Each NSD references to one or more VNFDs with their VLDs

**OPNFV:** OPNFV has established the Service Function Chaining project to develop functionality needed to provide VNF chaining in the OPNFV platform. The approach leverages on OVS, OpenDaylight and OpenStack integration and authorize OpenDaylight controller to control VNFs chains. Overall, the controller is responsible to manage Service Function Chain (SFC)

[^67]: [http://www.sonata-nfv.eu/sites/default/files/sonata/public/content-files/deliverables/SONATA%20D6.2%20Configuration%20of%20Pilots%20and%20pre-validation_v0.1_To%20be%20approved%20by%20EC.pdf](http://www.sonata-nfv.eu/sites/default/files/sonata/public/content-files/deliverables/SONATA%20D6.2%20Configuration%20of%20Pilots%20and%20pre-validation_v0.1_To%20be%20approved%20by%20EC.pdf)
topology, instantiate SFCs and associate them with Service Function Paths (SFPs), create and configure virtual routing and forwarding rules (VRFs), provide info including policy adopted for other SFC elements to properly interpret metadata etc. As the Technical Steering Committee has identified, the OPNFV VNF chaining architecture need also a VNF Manager to make a complete chaining solution. Thus, in Danube release, the OpenStack Tacker VNFM is used to create VM hosting VNFs and manage the life cycle of the last. Tacker will receive the ODL SFC configuration, manage the VNFs VMs, and forward the configuration to ODL SFC controller.

**Open-Baton:** Open-Baton is capable of managing the lifecycle of VNFs in a Network Service as per ETSI NFV MANO specifications. It does not support SFC itself, and it needs to be integrated with an SFC Orchestrator (through messaging queue system) and an SDN controller with SFC to implement the service function chaining features.

**SONATA:** SONATA follows a (network function and network service) descriptor based on ETSI official schema that is used for the deployment of the network service in the Service Platform. The descriptor provides information for all the Virtual Links (interconnecting VNF connection points), end-point information of the NS and the VNF Forwarding Graph that expresses the sequence and conditions for Service Function Chaining (SFC) to be enforced. According to the website, currently the NSD is still under consideration, mainly due to refinements in the service deployment and validations that are on-going.

### 3.2.2.3.10. Support VNFFG update during runtime

**OSM:** A tool called Network Service Scaling has been included on Release THREE. This tool has been defined as support for adding and/or removing full VNFs to/from a running Network Service. However, the main constraint is that he scaling action is triggered manually by the network operator using the UI or the Northbound API. Auto-scaling support is expected to be added in a future release.

**OPNFV:** Resource Management and dynamic update of VNF chains are not supported in the Danube release of OPNFV. Service function failover, load-balancing across multiple VNFs and dynamic service chain modifications by moving VNFs across multiple compute nodes are objectives of future releases (beyond Euphrates release which is planned for Oct. 2017)

**Open-Baton:** Open-Baton does not support VNFFG update at runtime.

**SONATA:** The support of VNFFG update during runtime is part of the work to be conducted until the end of the project.

### 3.2.2.3.11. Support hypervisor and container-based technologies

**OSM:** OSM needs Kernel-based Canonical’s LXD container technology configured on the host machine. Depending on the chosen VIM, different hypervisor and container technologies could be added.

---

**OPNFV**: OPNFV supports Kernel-based virtual machine (KVM) as a hypervisor to create virtual machines. It also integrates Kernel-based Canonical’s LXD container technology to virtualize the operating system and provide low-level container isolation mechanisms.

**Open-Baton**: Open-Baton does not impose any requirement on hypervisor to be used at VIM or to run the MANO application. The default VIM driver included in its release is for OpenStack, which can support multiple hypervisor and container technologies.

**SONATA**: As already mentioned above, SONATA supports OpenStack environment as NFVI for deploying VNFs in Virtual Machines, and all components comprising the Service Platform are based on Docker containers.

### 3.2.2.3.12. Background expertise

**OSM**: Telefónica (not Telefónica I+D) is member of OSM and NXW is participating in the project. UPM is trying to implement and make some tests.

**OPNFV**: In 5G-MEDIA consortium no strong expertise in OPNFV framework exist.

**Open-Baton**: NXW has strong expertise in Open-Baton framework mostly on the extension of the NFVO and the VNFM, during the SELFNET 5G-PPP project.

**SONATA**: UCL, leading the Work Package related to the Service Virtualization Platform is part of SONATA consortium and there is a strong interest from SiLO to promote this solution and be engaged in the sustainability of the SONATA platform in the framework of 5G-PPP.

---

Overall, the assessment showed that all MANOs have their own strengths and limitations. Particularly, OSM and SONATA are the two most prominent candidates under the considered criteria but none of them can be considered as a considerable better solution than the other. Nevertheless, OSM is supported by a larger community, which makes it a better candidate with respect to future sustainability plans. Despite that an advanced SDK toolbox is not currently exposed by OSM, there is noteworthy activity in integrating certain parts of SONATA’s SDK with it (e.g. the emulation platform), something that could be extended further to include more functionalities. In addition, TID plans already the integration between OSM and OnLife. In terms of all these, the OSM is selected the background MANO to be used in 5G-MEDIA platform.
4. **5G-MEDIA platform software architecture**

4.1. **High level architecture of 5G-MEDIA platform**

The logical architecture description of the 5G-MEDIA platform is shown in Figure 33.

![Figure 33 – Overview of 5G-MEDIA architecture](image-url)
In a top-down perspective, the architecture defines three layers of operations:

1) The DevOps layer, including the SDK and the OSS/BSS, which provides the means to service/apps developers and rest stakeholders to develop & deploy VNFs/NSs and access every exposed service by the SVP. The SDK provides the set of open source tools for supporting the rapid development of network applications (even on top of already existing VNFs, stored in the 5G-MEDIA VNF Repository) throughout the application/service lifecycle. It consists of proofing and packaging tools as well as emulator mechanisms to accelerate application development and provides a testing environment to be utilized prior to service deployment in the Service Virtualization Platform cloud resources. In the 5G-MEDIA project, SONATA SDK has been selected to drive the design of the SDK. Apart from enhancements over tools provided by SONATA SDK, one of the major project innovations is the integration of the serverless computing approach, leveraging open source projects such as OpenWhisk (while also introducing enhancements required to be the strict requirements of media applications). The main benefit from the integration of this paradigm is that developers do not need to care about the low-level details related to the infrastructure and operation specificities, thus drastically reducing development time and maintenance effort. Regarding the OSS/BSS, the consortium will evaluate possible available solutions and integrate the selected one in the 5G-MEDIA platform.

2) The SVP layer that hosts the components related to the OSM-based MANO framework (service & resource orchestrator, Infrastructure Manager(s), Repos etc.), as well as components of specific purposes, i.e. the 5G-MEDIA Catalogue and the Media Service Monitor Analyse Plan and Execute (MAPE) component. The core component of SVP is the MANO framework. Adopting the architectural principles of OSM, MANO functionalities are assigned over two main subcomponents in 5G-MEDIA SVP, i.e. the NFV MANO Service Orchestration and NFV MANO Resource Orchestration. The service orchestration sub-component undertakes responsibilities of NFVO and VNFM, while also the control of the VNF/NS Repository & Catalogue which is based on the innovation mechanisms achieved within the SELFNET 5G-PPP phase 1 project. The resource orchestration sub-component introduces a modular, customizable, and easily extensible plugin-based architectural model able to interact with multiple WIMs, SDNs and VIMs, including those enabling FaaS capabilities.

The 5G-MEDIA Catalogue, formally 5G Apps and Services Catalogue, is a new functional element which is designed to be NFV MANO platform-agnostic in terms of formats and syntax for NS descriptors and VNF Package information model. This catalogue uses a novel generalized and extendible format for representing NSs and VNFs, and it is capable to onboard NFV service elements from federated MANO systems (e.g. to complement a domain’s catalogue of NSs and VNFs with items made available by other federated domains), as well as Mobile Edge Cloud (MEC) media applications and services and other virtual applications such as SDN applications, and functions implementing the FaaS paradigm. Each request to the 5G-MEDIA Catalogue and SVP is verified by an AAA service responsible to identify end-users, grant authorized access to resources/services in case and keep records of activities in a consistent way.
A major innovation of the project is the development of the Media Service MAPE component, which is composed of the Cognitive Network Optimizer (CNO), the Monitoring service, the Planning and the Execution services. The Cognitive Network Optimization (CNO) Engine is taking advantage of the cognitive control principles developed in the COGNET phase 1 project of 5G-PPP to establish a ML-enabled optimization environment that dynamically establishes and updates the live VNF Forwarding Graphs (VNFFGs). To achieve this, it is driven by the monitoring service which aggregates various metric values of interest from every running application network service and integrated infrastructure (e.g. NFVIs). Apart from the CNO engine, these values are directly accessed through an open brokering system by the visualization tools of the SDK, as well as every internal service of SVP that may be interested in. The Planning service consists of different optimization models and caching strategies, linked with applications and tenants, supporting media and entertainment applications and their proper placement in NFVIs. Last, the Execution service triggers execution mechanisms according to the capabilities provided by OSM (i.e. scaling groups in NSDs) to enforce commands of the CNO over the integrated NFVIs and live VNFFGs.

3) The physical layer which is composed of every cloud computing, virtualization and other type of technology is used to host instantiated VNFs/NSs and deliver 5G-MEDIA application services to the end users. The purpose of the Core Network and Cloud environment is three-fold: 1) it provides sufficient resources to instantiate VNFs (or part of them in the microservice-based approach) that are used by multiple tenants or applications (e.g. virtual firewall), as well as application-specific helper functions/components (such as rendering and/or augmented reality servers), 2) it can be utilized to allocate resources following the network slicing concept in order to satisfy specific QoS/QoE requirements of an application or security/privacy concerns of a service provider, and 3) to facilitate the deployment of legacy components and services especially those instantiated on physical/specialized hardware (that is indeed a reality in media and entertainment applications development world). Several cloud-based edge networks and cloud environments will be connected to the SVP as NFVIs allowing for the instantiation of network applications closer to the user (edge computing paradigm). Particularly, the 5G-MEDIA SVP will be able to integrate several OpenStack- and VMWare-based environments by leveraging on VIM plugins already provided by OSM, while also OnLife and OW NFVIs by implementing corresponding specialized VIM plugins. Again, as the VNFs will be realized following the microservice approach, part of a network application might consist of several components deployed in more than one VIMs/NFVIs also connecting the end-users to the 5G-MEDIA ecosystem through mobile devices, tablets and other resource-constrained devices.

In the following subsections, the role and the main (sub)components and services of 5G-MEDIA software architecture are presented in more detail.

4.2. Service (Application) development kit

In general, an SDK is a set of tools that allows for the creation of applications, network services or functions and supports developers in implementing, packaging, deploying, monitoring or analysing the software. 5G-MEDIA SDK aims at providing a set of tools that helps developers
to easily implement and deploy new media related network applications to the SVP. In this scope, we are planning to reuse and extend open source SONATA SDK toolset and OSM design time tools in terms of 5G-MEDIA platform requirements. The SDK provides a programming model for application developers, which allows defining complex media services consisting of multiple VNFs. The high-level description of a first design for the SDK core components is shown in Figure 34.

As shown in Figure 34, the SDK is composed of the following components:

1) **Private Catalogue** consists of available VNFDs, NSDs and repository of VNF images.

2) **Validator** that is responsible for validating the VNFFGs in a few different ways, such as syntax, integrity or network topology. Validators expose special UIs to decrease developers effort to build a new error-less media applications or network services. Main Validators responsibilities are:
   - Syntax validation against schema definitions which are based on TOSCA Simple Profile for Network Functions and are aligned with ETSI MANO,
   - Integrity of connection endpoints in terms of the references of VNFs (name, vendor, version)
   - Detection of cycles and unconnected elements in the graph.

3) **Editor** is a set of UIs for developing, validating and onboarding media applications or network services to the 5G-MEDIA catalogue by building VNFFGs. In addition, Editors’
UI visualize topological dependencies or interconnection of involved VNFs of VNFFG and descriptions of individual VNFs. They also allow developers to add PNFs from catalogue to forwarding graphs.

4) **Emulator tool** supports developers to locally prototype and test complete network services in realistic end-to-end multi-PoP scenarios. This emulation platform allows the direct execution of real network functions, packaged as containers, in emulated network topologies running locally on the network service developer's machine. This environment can execute both a FaaS and a non-FaaS VNF. Figure 35 shows the scope of one of the candidate solution for emulation (SONATA/OSM Emulator a.k.a son-emu/vim-emu) and its mapping to the ETSI NFV reference architecture in which it replaces the NFVI and the VIM. However, the emulator tool can execute a FaaS VNF in the Lean OpenWhisk environment which is all in one development environment for serverless computing.

![Figure 35 – Mapping of son-emu to ETSI NFV MANO](image)

When the development and emulation stage is finished, the developer can deploy the tested FaaS VNF into production environment by using `wskdeploy` tool\(^{69}\) for pre-onboarding\(^{70}\) into the FaaS VIM of the production SVP. More specifically, artefacts previously pre-onboarded into the emulated FaaS VIM (implemented by Lean OpenWhisk) via `wskdeploy` as *managed project* can be later exported from the FaaS VIM as a project (the manifest of the project is created automatically) into a local file system where SDK is run and from there the project can be pre-onboarded to the production SVP (via `wskdeploy`) using the generated manifest.

5) **Monitoring tool** guides the developer by providing performance data of the media application and its components in a quantitative manner. This tool collects and

---

\(^{69}\) [https://github.com/apache/incubator-openwhisk-wskdeploy/blob/master/docs/export.md](https://github.com/apache/incubator-openwhisk-wskdeploy/blob/master/docs/export.md)

\(^{70}\) We refer to creating VNF artefacts in the FaaS VIM (creation of an action in FaaS VIM, i.e., OpenWhisk repository logically corresponds to pre-loading a VNF image to e.g., Open Stack VIM) prior to onboarding of a VNF via MANO as *pre-onboarding*. See Deliverable D3.2 for more details.
visualizes network monitoring data of a service. Depending on the SVP monitoring functionalities, a generic monitoring tool includes both resource consumptions metrics of a VNF (i.e. CPU, memory, etc.) and VNF specific traffic analysis metrics (i.e. hit ratio etc.).

6) **Profiling tool** allows developers to stage a service in a local SDK environment to overcome possible issues related to either configuration or implementation of a service before deploying in production in SVP. The performance of NFV based services including VNF implementations relies on different metrics such as the underlying software platform (e.g. programming language, compiler etc.), implementation quality, the architecture of underlying hardware platform (e.g. memory, number of cores, storage etc.), the potential variability of the interconnecting networks between VNFs due to their flexibility of deployments on different locations in the network. Measuring these performance metrics of a VNF or a NS under different set of software, hardware and network conditions is called offline profiling. Developers benefit from offline profiling of VNF implementations to tune the required system resources in production deployments.

7) **Packaging tools** among those developed within the “Cloud computing” community, open source tools such as Vagrant, Chef, Puppet, Docker are going to be evaluated and integrated in the SDK as part of the Operation Support Tools in a seamless manner. Following a DevOps model, these tools collect all the necessary software artefacts (e.g. required libraries) and generate software packages in the form of lightweight containers or virtual appliances to be integrated in the Private/Public Catalogue.

8) **Platform Catalogue** consists of available VNFDs, NSDs and repository of VNF images on the SVP.

The detailed technical specifications of each component are presented in “D5.1 - 5G-MEDIA Programming Tools”71.

### 4.2.1. Cross reference of 5G-MEDIA SDK to SONATA SDK

Table 5 summarizes the main features and services of each core component of 5G-MEDIA SDK and highlights its innovation and advances with respect to the corresponding component of the SONATA SDK or OSM Design Time Tools (whenever applicable). Thus, 5G-MEDIA SDK aims to allow developers to benefit from 5G-MEDIA platform functionalities and services. In order to achieve this aim, 5G-MEDIA SDK focuses following enhancements of the 5G-MEDIA Platform, namely FaaS development and emulation, OSM compliance, media application development use-cases and CNO policy rules etc.

---

71 [http://www.5gmedia.eu/outcomes/deliverables/](http://www.5gmedia.eu/outcomes/deliverables/)
### Table 5 – Services and innovation of 5G-MEDIA SDK in contrast to SONATA SDK and OSM Developer Tools

<table>
<thead>
<tr>
<th>Component</th>
<th>5G-MEDIA</th>
<th>SONATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Catalogue</td>
<td>• Private repositories serving individual service developers.</td>
<td>Storage service for available VNFDs, NSDs and repository of VNF images.</td>
</tr>
<tr>
<td>Editors</td>
<td>• All in one UI (all SDK tools)</td>
<td>Uls for developing, validating and onboarding network services by building VNFFGs.</td>
</tr>
<tr>
<td></td>
<td>• FaaS Emulation UI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Application developer specific UI</td>
<td></td>
</tr>
<tr>
<td>Emulators</td>
<td>• FaaS Emulation</td>
<td>Emulation service for network services in realistic end-to-end multi-PoP scenarios.</td>
</tr>
<tr>
<td>CLI Tools</td>
<td>• FaaS CLI Tools</td>
<td>CLI tools for developing, validating and onboarding network services.</td>
</tr>
<tr>
<td>Monitoring Tools</td>
<td>CNO Policy Rules Results’ Visualization</td>
<td>Monitoring service for performance data of the media application/network service and its components in a quantitative manner</td>
</tr>
</tbody>
</table>

#### 4.2.2. DevOps continuous integration and deployment services for the SDK

The 5G-MEDIA SDK interacts with the continuous integration and deployment services to provide building automation, support to multiple ISO formats (e.g. plain ISO, unikernel, Docker) and facilitate the image uploading on the NFVI according to DevOps best practices to get shorter development/operation cycles.

Such services are mainly a source repository to keep track of software artefact source code (Gitlab), an artefact repository to store libraries, their version and manage dependencies (Nexus), source code quality (Sonarqube), a central image repository to store VDU images (plain ISO, Docker containers, unikernel images, etc.) and a service to build them (Jenkins) that triggers automatically after each commit on the source code repository. These services have been described in the deliverable “D2.1 – APIs and Tools for Operation”.

The typical use case in which the developer uses the source code repository as a common base for its code and to enable concurrent development in teams; in the context of 5G-MEDIA project it can also been specified which format should be created for each ISO that is going to be the VDU for the VNF under development. A dedicated CLI tool generates and customize the templates necessary to support the compilation by the CI/CD tools and puts the resulting ISO on the central image repository, available for the testing phase on the SDK test environment, before being considered production ready. As a consequence, such ISOs are then referred

---

respectively from the private catalogue during the development/testing phase, and the public
catalogue once such VNFs are ready to be used in production.

Figure 36 shows the main CI/CD services and their relation with the SDK for DevOps; the
interactions described above will be described in the following section about sequence
diagrams.

![Figure 36 – SDK and main CI/CD services for DevOps](image)

4.3. Service virtualization platform and core components

4.3.1. 5G-MEDIA MANO framework

As already mentioned, the 5G-MEDIA project leverages on Open Source MANO (OSM)\(^73\) to
meet the requirements of NFV/SDN network, aligned with ETSI NFV model. Thus, the two main
subcomponents of MANO framework, shown in Figure 33, are the Service Orchestration and
the Resource Orchestration, respectively. Besides those, 5G-MEDIA SVP also introduces two
other components of critical role, i.e. the 5G-MEDIA Catalogue and the MEDIA Service
Monitoring, the workflow and responsibilities of which are presented in the following
subsections.

4.3.1.1.NFV MANO Service Orchestration

The Service Orchestration sub-component performs all aspects of service orchestration,
including VNF/NS lifecycle management and end-to-end, resource-coordinated services

\(^73\) [https://osm.etsi.org/](https://osm.etsi.org/)
execution in an otherwise dispersed NFV environment. Its two main modules are the NFVO and the VNFM.

1) NFVO coordinates, authorizes, releases and engages NFVI resources among different PoPs or within one PoP. This is done by engaging with the VIMs (e.g. through a north bound APIs provided by the last) instead of engaging with the NFVI resources, directly. In terms of service orchestration, NFVO oversees the allocation of resources to NSs, monitors the allocation needs for the instantiated NSs, coordinates their parts (e.g. VNFs, connectivity information between them etc.) in runtime and controls their topology management and update. It achieves these by interacting with every active VNFM (so it does not need to talk to VNFs directly) and the Media Service MAPE (so to update VNFFG in runtime). In summary, NFVO is responsible to:

- Control NS lifecycle management (including instantiation, scale-out/in, performance measurements, event correlation, termination) and global resource management.
- Validate and authorize network functions virtualization infrastructure (NFVI) resource requests.
- Apply VNFFGs optimization of CNO engine, work with other blocks to optimize network orchestration and keep an end-to-end view of NSs and integrated environments statuses.

2) Each VNFM (VCA layer in OSM terminology) has the responsibility to oversee lifecycle management of VNF instances (creation, deletion and scaling up/down VNF resources) and coordinate and adapt configuration and event reporting between corresponding NFVIIs and the EMS. Specifically, the VNFM is responsible for the FCAPS operation over VNFs (i.e. Fault, Configuration, Accounting, Performance and Security Management).

4.3.1.2. NFV MANO Resource Orchestration

The Resource Orchestrator is responsible for managing and coordinating resource allocations across multiple geo-distributed VIMs and multiple SDN controllers. It exposes a northbound API to communicate with the Service Orchestration sub-component and provide a number of utilities for internal consumption. In line with the specifications of OSM, the Resource Orchestrator adopts a plugin programming model, which allows to add functionality without modifying or having access to its source code. Each plugin is responsible to connect the interface of the corresponding entity with the Resource orchestration. The four types of plugins that are supported are:

1) VIM plugins: Each integrated VIM is responsible to control and manage the compute, storage, and network resources within one operator’s infrastructure sub-domain.

---

74 The complete list of operations addressed by the NFVO is provided in “D31 – Initial Design of the 5G-MEDIA Operations and Configuration Platform”.

75 First specifications and operations performed by the VNFM are presented in “D31 – Initial Design of the 5G-MEDIA Operations and Configuration Platform”.
5G-MEDIA SVP, there may be multiple VIMs where each one manages an individual infrastructure domain. Under this scope, each VIM interacts directly with the corresponding NFVI domain to deploy NSs therein and manage the available resources. Thus, every VIM implementation should be able to maintain an inventory of the physical resources and also keep track of their utilization and their map to virtualization resources. Apart from OpenStack and VMWare plugins, which are already provided by OSM release THREE and beyond, the 5G-MEDIA SVP will also interact with OnLife NFVI by delivering a specialized VIM plugin for OpenNebula.

2) SDN plugins: Each integrated SDN controller undertakes the traffic/flow control throughout the underlying network elements to enable intelligent and efficient networking. 5G-MEDIA will make use of one or a combination of OpenDaylight, ONOS and Floodlight plugins which are supported by the OSM stack.

3) FaaS plugins: Each FaaS plugin integrates serverless computing capabilities into the 5G-MEDIA platform. The 5G-MEDIA project will deliver a plugin to integrate FaaS model implemented by OpenWhisk into the SVP.

4) WIM plugins: Each WIM plugin abstracts the interactions between multiple WANs over the which the VNFs/NS may be instantiated.

4.3.2. The 5G-MEDIA catalogue

The 5G-MEDIA Catalogue, named 5G App and Service Catalogue, is a fundamental part of the 5G-MEDIA SVP and the 5G Management and Orchestration architecture in general. In fact, it provides all the functionalities needed for CRUD operations on NS and VNF Packages, i.e. onboarding, update, delete and information fetch. The catalogue design is based on the concept of a novel generalized and extensible format for describing NSs along with the VNFs composing them. This catalogue leverages on the design and development background on the SELFNET Catalogue and the latest ETSI NFV MANO specifications 76, 77, 78. The catalogue allows to handle also the onboarding and management of other types of descriptors/packages in the IT domain, like MEC applications, SDN applications and functions developed through FaaS paradigm. The first motivation behind the choice of adopting a generalized format for descriptors is the high fragmentation that currently affects the way applications and services are represented across different MANO framework: this is de-facto impacting the ability of


77 ETSI Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification, SOL004, Available online: http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.04.01_60/gs_NFV-SOL004v020401p.pdf

78 ETSI Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point, Available online: http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf
NFV developers and service providers to produce a portable offering of their virtual applications and services. To date, ETSI NFV MANO frameworks available in state of the art (e.g. ETSI OSM, Open-Baton, Tacker, Riff.Ware, etc.) adopt proprietary – though ETSI standards inspired – approaches: different formats (e.g. JSON, YAML, etc.) are used to describe NS-es and VNFs, packages contain different information (e.g. monitoring parameters, software images, etc.) and different procedures are used for specifying and configuring service components (e.g. JuJu charms, cloud-init, other Day1/Day2 configuration recipes, etc.). It is common practice that NFVO specific descriptors and packages are created for each of the different MANO platforms used by service providers, with custom mechanisms and approaches to encode variables in descriptors (e.g. for VF configuration, monitoring, generation of outputs), to encode auto-scaling rules, to support NFVI specific extensions (e.g. SR-IOV, DPDK, containers instead of VMs, etc.). Moreover, most of the current deployments are in a context of single MANO domain operations, while end-to-end 5G networks will be composed of multiple virtualized infrastructures, split in different administrative domains.

New ways are needed in more complex 5G deployments for composing services from different providers (e.g. using SDKs) as well as new mechanisms are called by service providers for orchestrating and delivering NS which are not limited by the characteristics and capabilities of a single provider’s NFVI but can more flexibly extend to cover larger geographical coverage (beyond the single data-center), a higher amount of resources available both at the edge and at the core of the network. The Vertical Application developer (i.e. the customer of the NFV Network Operator) should have the possibility to onboard new virtual applications for his specific services, while the service provider should be able of using Virtual Functions from federated MANO systems (i.e. complement my domain catalogue of NS-es and VNFs with items made available by other federated domains).

Based on these various requirements and the limitations of state of the art NFV catalogue solutions, a new 5G App and Service Catalogue is designed to be NFV MANO platform-agnostic in terms of formats and syntax for NS descriptors and VNF Package information model. The new format for representing Network Services and Virtual Network Functions leverages on the latest ETSI NFV standards for NS/VNF descriptors 79, 80, VNF packages81 and the MEC

79 ETSI Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification, IFA014, Available online: http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.04.01_60/gs_NFV-IFA014v020401p.pdf


81 ETSI Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification, SOL004, Available online: http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.04.01_60/gs_NFV-SOL004v020401p.pdf
Application descriptors\(^{82}\). The high-level design of the 5G App and Service Catalogue is depicted in Figure 37.

![Figure 37 – 5G App and Service Catalogue high level design](image)

This design reflects the modularity and the adaptability of the NS and VNF/VA information models to be adopted. In fact, The North-Bound Interface of the catalogue is designed to support the NS Management and VNF Package Management interfaces as specified in the ETSI NFV specification for the Os-Ma-Nfvo reference point\(^ {83}\). The Admin Interface, instead, is intended to offer functionalities for the management of users with configuration of the related policies. The South-Bound Interface of the Catalogue is composed of different plugins capable of handling the translation of the generalized package/descriptor into the specific format expected at the underlying orchestrator (both NFV and SDN orchestrators could be supported).

---

\(^{82}\) ETSI Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management, MEC 010-2, Available online: [http://www.etsi.org/deliver/etsi_gs/MEC/001_099/01002/01.01.01_60/gs_MEC01002v010101p.pdf](http://www.etsi.org/deliver/etsi_gs/MEC/001_099/01002/01.01.01_60/gs_MEC01002v010101p.pdf)

\(^{83}\) ETSI Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point, SOL 005, Available online: [http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf)
and actuating onboarding/management operations on the target virtualization platform. In particular, the MANO plugins include:

- A translation module responsible for translating the generic descriptor/package in the format expected at the underlying MANO Service Orchestrator (e.g. packages compatible with the OSM information model specification),
- A set of VIM plugins (e.g. OpenStack plugin, OpenWhisk plugin etc.), one for each VIM in the NFVI administrated by the target MANO stack, for uploading images in the VIM images’ storage,
- A MANO agent for collecting feedbacks about onboarding and instantiation operations as well as for notification about, for instance, new VIM instances or new capabilities supported by the MANO framework.

The 5G App and Service Catalogue design includes also a Notification Dispatch Interface for distributing service and application specific notifications to a set of consumers listening on a notification bus (e.g. based on Kafka message queue and streaming platform). In 5G-MEDIA, a specific consumer on the message bus is the MAPE component, which retrieves application specific monitoring parameters used to initiate monitoring jobs once the service/application is instantiated through the MANO stack.

### 4.3.3. Media service MAPE

The MEDIA service MAPE component is responsible to track versatile metrics about the performance of the instantiated VNFs/NSs and application services, as well the status of assigned resources in both computation and networking terms. These metrics are used for network resource optimization, dynamic VNFFG adaptation, NS monitoring and QoS/QoE guarantee. The MEDIA service MAPE is composed of two main sub-components that realize the MAPE concept, the Monitoring Service and the Cognitive Network Optimization Engine. The role and the mission of both these sub-components is explained in the following two subsections, while their detailed technical specifications are provided in “D3.3 - Specification of the 5GMEDIA QoS Control and Management Tools”.

#### 4.3.3.1. Metrics acquisition, monitoring and sharing services

In 5G-MEDIA, the monitoring service aims to meet the following specifications:

1. It must be able to collect different types of data for every instantiated NSs/VNFs orchestrated by the MANO framework of SVP, including CPU usage, memory allocation, active/passive network throughput, packet loss, etc.
2. It must be able to track data for VNFs and NSs instantiated over every virtualization environment (e.g. VM-based, container-based) and underlying NFVI (e.g. OpenStack, VMware, OnLife, etc.) which is supported by the 5G-MEDIA SVP.
3. It must be able to collect data from the SDN controller, its control-data interfaces and used protocols.
4. It must drive operation of CNO, as well visualization/monitoring services of SDK that allow developers/service providers/platform operators to monitor behavior of deployed NSs (Monitoring as a Service).
5) It must be able to accommodate VNF-specific and NS-specific alerting rules for offline and real-time notifications directed not only to internal functional blocks (e.g. NFVO) but also the external actors (e.g. developers, service providers).

6) It must ensure that the monitoring services work in unison with MANO functional blocks and provide them any monitoring information they need to fulfil their management and orchestration tasks.

7) It must be able to meet near real-time responsiveness, accessibility and scalability, and ensure fault tolerance mechanisms that do not permit for downtime due to failed hardware or software modules.

The high-level description of the data acquisition, monitoring and sharing subsystem is shown in Figure 38.

---

**Figure 38 – The architecture of data acquisition and sharing in Media Service MAPE**

The subsystem collects several metrics to track the condition of resources in the integrated NVFI environments, the performance of the network and the behaviour of the media traffic at the application layer. To share metrics in (almost) real-time among the internal services of the media service MAPE, a pub/sub brokering mechanism has been deployed by using open-
source Apache Kafka\(^{84}\). Also, the metrics are stored in a database for future reference. Confidentiality and integrity of exchanged data in the bus, as well as authentication and authorization of pub/subs clients are in line with the 5G-MEDIA AAA services. The management and the configuration of the Kafka bus, the considered data models in messages exchanges, the traffic streaming policies, as well the types of metrics which are collected per environment (either NFVI or the application layer) are thoroughly presented in D3.3 “Specification of the 5G-MEDIA QoS Control and Management Tools”\(^{85}\). Note that the Kafka bus is open to all the internal services or entities of the 5G-MEDIA SDK and SVP that may interest to access, retrieve and consume either stored (in the database) or current metrics from the tracked environments. Overall, the major types of Kafka clients that prosumne data over/to the Kafka broker are:

- Various data adaptation and transformation services to push data from the monitored environments (exporters), enrich the collected metrics and prepare their proper consumption by other services of the media service MAPE, SVP and SDK.
- A data base that keep records (history) of collected metrics (either raw or processed) to address future reference by the services of the SVP (e.g. the accounting/billing services etc.).
- The ML and policy/optimization engines of CNO aim to optimize VNF placement, network performance and media service deployments.
- The monitoring services of SDK, e.g. dashboards and visualization tools, offered to the end-users (e.g. service providers, developers etc.) for tracking the behavior and the performance of the instantiated NSs.

The detailed technical specifications of the data acquisition, monitoring and sharing subsystem is provided in D3.3 “Specification of the 5G-MEDIA QoS Control and Management Tools.

### 4.3.3.2. Cognitive network optimization engine

The reference control model for resource optimization and dynamic VNFFG adaptation in 5G-MEDIA is shown in Figure 39. The overall architectural work is based on the following four steps:

\(^{84}\) [http://kafka.apache.org/](http://kafka.apache.org/)

\(^{85}\) [http://www.5gmedia.eu/outcomes/deliverables/](http://www.5gmedia.eu/outcomes/deliverables/)
The overall architectural work is based on the following four steps:

- **Monitoring and data collection**: Information from running VNFS, the NFVI and networking environment is gathered by the data collector – more details on the monitoring system are given in section 4.3.3.

- **Analysis and prediction**: The information gathered by monitoring is processed by the machine learning engine to predict/forecast future trends in user demand, network conditions and resource availability. The intelligent forecasts become the input for the policy/optimization component.

- **Resource allocation/planning**: By using optimization techniques such as integer linear programming and/or greedy heuristic algorithms, this component is responsible for devising a plan to efficiently use resources in order to achieve system goals. During the project we are considering two different cases for optimization algorithms with different targets:
  - First, are instructions to the MANO to instantiate/tear-down and scale out/in NSs, VNFS and VNFFGs. These actions will optimize the deployment and usage of the media NS and its component VNFS.
  - Second, are instructions to alter the behavior of the application logic deployed in the VNFS themselves. Examples include the configuration of compression parameters in vTranscoders or the load splitting ratios to be implemented by application-specific load balancing functions.

- **Execution**: The MANO Service Orchestration components send instructions to the RO components to instantiate and configure the VNFS in the underlying NFVIs to meet the new optimization objectives determined by the previous phase.
One of the main components of the CNO is the Policy/Optimization component, shown in Figure 40. A range of options exist for implementing the optimisation algorithms. For instance, Integer Linear Programs can be used for small input datasets in which an optimal solution can be obtained within an acceptable time. Alternatively, suitable heuristic algorithms can be used to find close-to-optimal solutions for larger input datasets. A third option is to implement the optimisation decisions using machine learning techniques rather than using machine learning solely as a means to forecast demand, which is then input into traditional optimisation functions. The optimization model will be programmable to follow policies defined by the service provider. For example, the service provider can determine the weight that the optimisation algorithm gives to reducing costs versus improving performance, or it may define the maximum cost budget for any particular solution, or the maximum latency acceptable for its users.

The interaction of the Policy/Optimization with the NFV MANO Service Orchestrator is shown in Figure 41. At instantiation time, the service provider triggers the VNF placement component of NFVO to pre-deploy VNFs and make them available before user requests arrive. At demand time, user requests will be served by the VNF forwarding graph management component. This is done by redirecting user requests to the best ingress point of VNFFGs. The policy/optimization component executes periodically to optimize VNF placement and VNF forwarding. On the other hand, when a user request cannot find an appropriate VNFFG, it will trigger the policy/optimization component to instantiate VNFs to serve that request.

It should be noted that the dynamic management of VNFFGs, including the instantiation of VNFs and the construction of VNFFGs tailored to individual user requests, is an advanced NFV feature which is not currently supported by existing MANO implementations. While the opportunities for optimisation of media services through fine-grained manipulation of VNF scaling and the mapping of user requests to VNFFG instances will be studied theoretically, the degree to which it will be possible to implement and test these aspects using off-the-shelf MANO software will be determined in the second year of the project.
4.4. Extensions in virtualized infrastructure managers

The 5G-MEDIA project plans to deliver full compatibility between its MANO framework and two popular cloud management and apps orchestration platforms and their corresponding VIMs, i.e. the OnLife platform (using OpenNebula as VIM) and OpenWhisk (using FaaS VIM).

4.4.1. Virtualized infrastructure manager for OnLife

The reference implementation of OnLife/CTpd relies on OpenNebula 5.0 VIM. In the following list we detail the design decisions made to consider the specific operational requirements of OnLife/CTpd:

- **Virtualization:** The compute nodes use KVM as hypervisor. Guests are configured to use Virtio device drivers with the host page cache disabled.

- **Storage:** The applications running in CTpd are considered stateless to easily and rapidly provision them. Golden images are prepared and parametrized for each service and specialized for each customer at boot time. OpenNebula is configured to store disk images in the local storage.

- **Networking:** NFV traffic is signaled by VLAN tags, this greatly simplifies the network interconnection of the VMs with the Clos fabric and allows us to use Linux bridges. OpenNebula is responsible to setup the VLAN tags and bridges for each VM.

- **Monitoring and Introspection:** OpenNebula monitoring stack gathers raw performance metrics (e.g. CPU or bytes transmitted/received by the VM). This information will feed Telefonica's internal alert and monitoring system. Additionally, VMs push custom service related metrics to OpenNebula to trigger scalability rules or alerts.

Figure 42 shows a screenshot of the OpenNebula GUI working in the CTpd PoC.
In terms of networking, ONOS is the selected SDN platform for the deployment of CTpd, as most of CORD functionality relies on applications developed in this platform. More specifically, CTpd focuses on the R-CORD use case, which involves the use of the vOLT and the vRouter applications, already developed for ONOS. Further deployment into the Enterprise and Mobile segments will follow the same approach as the access network becomes one and the same for all segments. Other reasons of choosing ONOS are its simple and dynamic deployment via the Apache Karaf environment, its high-quality documentation and its community support.

In 5G-MEDIA, the integration between OpenNebula and OSM will be developed using a plugin allocated at the OSM’s RO component similar to other cloud platforms, as shown in Figure 43. The objective is that OpenNebula behaves as a VIM additionally to the official VIMs supported Out-of-the-Box by OSM.
The interface used to interact with OpenNebula in OSM is its XML-RPC API through a python library created. XML-RPC is the primary interface for OpenNebula, exposing all the functionality to communicate with the OpenNebula daemon. Through the XML-RPC interface you can control and manage any OpenNebula resource, including VMs, Virtual Networks, Images, Users, Hosts and Clusters. It will allow a low-level interface with OpenNebula core. In Figure 45, it is shown OpenNebula system interfaces and that the XML-RPC API is very close to the core.

In 5GMedia the only way to interact with opennebula is through OSM that will use the XML-RPC API, but as we can see there are much more other interfaces separated in two categories:

- **Cloud interfaces**: are primarily used to develop tools targeted to the end-user, and they provide a high-level abstraction of the functionality provided by the Cloud. They are designed to manage virtual machines, networks and images through a simple and easy-to-use REST API. The Cloud interfaces hide most of the complexity of a Cloud and are especially suited for end-users.

- **System interfaces**: expose the full functionality of OpenNebula and are mainly used to adapt and tune the behavior of OpenNebula to the target infrastructure

Thus, when a service is created—for instance, Internet access or a CDN, a VLAN is assigned to it and the VMs needed to implement the associated virtualized network functions are deployed. Internally in OnLife, OpenNebula and ONOS are responsible for a different area: ONOS takes care of the network forwarding and OpenNebula of the traffic delivery within each compute node. In this way, ONOS sets up the flows for the Clos switches to route network traffic to the target node. The traffic is signaled with VLAN tags for each networking service, OpenNebula sets up specific bridges and tags interfaces to connect each VM in the node to
the target network. Further, VM deployment and resource allocation is performed by OpenNebula, which interfaces with ONOS through the CLOsfwd application to register the new VM, so routes in the Clos can be created to connect the service VMs. The SDN and cloud layer integrate using a special network driver in OpenNebula that performs the following operations:

- VM deployment: The driver creates a device tag for the service and a Linux bridge to attach the VM to. In this phase it also registers the new VM within the CLOsfwd ONOS service, so the corresponding routes can be installed. The driver sends the OpenFlow device and port for the selected compute node, and the VLAN ID to the CLOsfwd application.
- VM termination: The driver deletes the associated networking devices in the host and invokes the CLODfwd application to remove the service flow.

4.4.2. Virtualized infrastructure manager for OpenWhisk

Figure 45 provides a high-level description of FaaS integration with the ETSI MANO stack. FaaS technology is transparent to the upper layers of the stack, VNFM and NFVO. The integration with the stack occurs at the level of VIM. On the northbound interface, there are no changes to the VIM API to maintain compatibility with the ETSI MANO standard. On the southbound interface, FaaS VIM is FaaS technology specific.

In the 5G-MEDIA reference implementation, we use OSM as Resource Orchestrator and Apache OpenWhisk as the FaaS technology. We implement an OSM plugin to realize a concrete FaaS VIM. The southbound interface of the FaaS VIM reference implementation is OpenWhisk specific. It is easy to see that this way, we can easily incorporate other serverless technologies, such as Open Lambda[^86], OpenFaaS[^87], Kubeless[^88] and Fission[^89]. In principle, in the future, 5G-MEDIA platform can also incorporate close source serverless technologies, using this pluggable architecture approach.

[^86]: https://open-lambda.org/
[^87]: https://www.openfaas.com/
[^88]: https://kubeless.io/
[^89]: https://fission.io/
Figure 45 – FaaS Integration with MANO

Figure 46 shows the reference software architecture of the FaaS VIM in 5G-MEDIA. In the implemented architecture, the FaaS framework can use a container orchestration engine installed either on top of the bare metal machines or on top of the virtual machines provisioned and managed by a virtualization management layer. In our reference implementation, we use Kubernetes as our container orchestration engine (the reason for selecting this specific technology is that Kubernetes became a de facto standard for container orchestration), OpenStack and OpenNebula are used as virtualization technology in UC1 and UC2 settings, respectively.

---

90 https://kubernetes.io/
Figure 46 – Reference software architecture for FaaS VIM

Figure 47 describes a reference implementation of UC1 using FaaS VIM, OpenWhisk, Kubernetes, and OpenStack. The green rectangles depict containers running VNFs that are instantiated via the FaaS (OpenWisk) VIM in response to the gaming application instantiation request. All VNFs are instantiated together upon a gaming session creation except vReplay. The latter is instantiated on demand from the gaming server when a replay clip should be created from the media buffered by the vBuffer VNF.

Figure 47 – FaaS/MANO sample software architecture

The VNF containers are interconnected by the Flannel network (green), which is an overlay on the network provided by OpenStack’s Neutron on top of the physical network (blue). Note, that in a general case, VNFs do not execute in the same Kubernetes instance. Moreover, they might execute in the different Data Centers. In this case, the VNF containers will be exposed
as external services via the Kubernetes Ingress\textsuperscript{91}. For more details on FaaS networking setup consult Deliverable D3.2.

### 4.5. Sequence diagrams for selected operations

This section makes use of UML sequence diagrams to present high-level workflow of the most critical services/actions taking place in the 5G-MEDIA ecosystem. The considered services are shown in Table 6.

#### Table 6 – Main services/actions in the 5G-MEDIA ecosystem

<table>
<thead>
<tr>
<th>#</th>
<th>Service/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VNF development cycle</td>
</tr>
<tr>
<td>2</td>
<td>NS emulation in SDK</td>
</tr>
<tr>
<td>3</td>
<td>VNF package onboarding in the Catalogue</td>
</tr>
<tr>
<td>4</td>
<td>FaaS development cycle</td>
</tr>
<tr>
<td>5</td>
<td>NS onboarding in the Catalogue</td>
</tr>
<tr>
<td>6</td>
<td>NS instantiation</td>
</tr>
<tr>
<td>7</td>
<td>NS scale-out running FaaS</td>
</tr>
<tr>
<td>8</td>
<td>VNF forwarding graph optimization in runtime</td>
</tr>
<tr>
<td>9</td>
<td>NS monitoring in MAPE loop</td>
</tr>
<tr>
<td>10</td>
<td>Catalogue authentication and authorization</td>
</tr>
<tr>
<td>11</td>
<td>Accounting</td>
</tr>
</tbody>
</table>

For each action shown in Table 6, the depicted sequence of activities focuses on the perspective of the entity which has the main responsibility of the demonstrated action/service.

#### 4.5.1. VNF development cycle

Figure 48 shows the image packaging of an application. When an application developer pushes new commits to the applications’ source code repository (e.g., Gitlab), 5G-MEDIA CI/CD pipeline packs and uploads the application image (e.g, Docker, unikernel, plain VM iso, etc.) based on the given configuration file (e.g, Docker configuration file, unikernel template, etc., Vagrant file, etc.).

\textsuperscript{91} https://kubernetes.io/docs/concepts/services-networking/ingress/
4.5.2. NS Emulation in SDK

Figure 49 shows the general architecture of emulator platform (son-emu/vim-emu and lean-openwhisk) and depicts the high-level workflow of a developer using it. First, the application developer defines a NS package, consisting of service’s NSD and functions’ VNFDs, as well as...
docker files or pre-build images that contain the network functions to be tested (1 and 1.1). Second, the developer defines a multi-PoP topology on which he wants to test the service (2) and starts the emulator with this topology definition (3). After the emulator has been started, the developer connects the OSM (or the MANO stack of his choice) to the emulated PoPs in the platform by using standard cloud interfaces (4). Now, the application can be deployed on the platform by pushing it to the MANO system (5) which starts each network function as a docker container, connects it to the emulated network, and sets up its forwarding chain. Finally, the service is deployed and runs inside the platform (6).

4.5.3. VNF package onboarding in the Catalogue

Figure 51 shows the procedure to onboard a VNF Package into the 5G App and Service Catalogue and, consequently, into the NFV Orchestrator catalogue in the underlying SVP. This onboarding procedure is made performing fourteen steps:

1. The first step is the creation of the VNF Package resource Id in the 5G App and Service public Catalogue.
2. The Catalogue creates the resource Id.
3. The resource Id is created and returned back to the sender.
4. The sender requests the onboarding of the VNF Package using the resource Id already created.
5. The Catalogue notifies to the sender that the onboarding request has been received.
6. The Catalogue processes the onboarding request performing different internal actions, in particular:
   a. Checks that all mandatory elements are included.
   b. Validates the VNF Package integrity and authenticity by checking manifest file and manifest file security.
c. Translates the VNF Package from the generalized format expected at the Catalogue NBI into the specific NFVO format expected at the underlying SVPs.

7. The Catalogue requests the uploading of the image(s) at the VIM(s) in the NFVO administrative domain.

8. The VIM(s) notifies to the Catalogue that the uploading requests has been received.

9. The Catalogue starts a polling task for retrieving the status about the uploading of the image(s).

10. The status about the uploading of the image(s) is retrieved by the Catalogue, which re-iterate the polling task until the uploading is completed.

11. The Catalogue requests the onboarding of the translated VNF Packages at the underlying NFVO catalogues.

12. NFVO notifies to catalogue that the onboarding request has been received.

13. Once the onboarding is done, NFVO notifies to the Catalogue the operation result.

14. The Catalogue notifies to the Sender the VNF Package onboarding operation result once all the onboarding/uploading operation at the underlying MANO platforms are completed.
4.5.4. FaaS development cycle

Figure 52 shows the onboarding message flow for a VNF using FaaS. The main difference from the VM based VNF package onboarding is in Step 4 when the FaaS based VNF “image” is pushed into the FaaS framework repository (e.g., OpenWhisk). There is no “image” in the regular sense. Rather, there are artefacts, such as actions, triggers, rules packages that together comprise the FaaS based VNF. As described above, when instantiated, OpenWhisk actions execute as containers. For standard actions the containers images are pre-built and only the code and metadata are stored in the OpenWhisk repository. The artefacts are put to the OpenWhisk repository using wskdeploy tool of the SDK as described earlier in this document.

In case of black box actions, the containers can be stored either in some container registry, e.g., in Docker-Hub and pulled on demand using the metadata stored in the OpenWhisk repository.
Figure 52 – On-board VNF FaaS package flow

In the OSM environment, the flow would be slightly different. During the on-boarding phase, VNF images are not uploaded. Rather, they should be pre-uploaded into the FaaS VIM repository before the VNF onboarding. We term this phase “pre-onboarding”. In Figure 53, we depict the pre-onboarding process for FaaS VNFs. 5G-MEDIA comprises both a centralized cloud and distributed edges. In the federation architecture for OpenWhisk developed in 5G-MEDIA, OpenWhisk instances can serve in the roles of a “leader” or a “follower”. The leader OpenWhisk has a state of the OpenWhisk assets (e.g., actions). Suppose a user has developed a container based “black box” OpenWhisk action implementing a VNF. For the sake of the example, we refer to this VNF as vTranscoder. A user pushes the VNF container to a public repository (Step 1) or a private one (Step 1’). Next, the user defines a project using wskdeploy tool (Step 2). Let’s assume that the project name is “game-project”. This step creates a metadata about the asset in the database of the OpenWhisk leader. In Step 3, the user invokes a special management action that marks the previously created project with the target followers belonging to the type “game”. The asset (the VNF action) created in steps 1-3 can now be explicitly propagated to the follower OpenWhisk instances by either calling a special management action explicitly (Step 4’) or it can be left to the federation protocol that will request the state from the leader automatically on a periodic basis (Step 4) or will be pushed by the leader (Step 5). It should be stressed that the followers and the leader are independent autonomous OpenWhisk instances. The assets across these instances will be kept eventually consistent by the leader/follower federation protocol in spite of intermittent connectivity and node failures.
With the exception of the FaaS specific pre-onboarding and on-boarding, the standard ETSI MANO flows are unchanged. Hence, we omit them in this document. For further details consult Deliverable D3.2.

### 4.5.5. NS onboarding in the Catalogue

Network service on-boarding is the process which allows to submit an NSD to the NFVO in order to be included into the catalogue. This process is presented in Figure 54 and its steps are:

1. The Sender requests the creation of a new NSD resource Id to the Catalogue.
2. The Catalogue creates the NSD resource Id.
3. The resource Id returned to Sender.
4. The NSD is submitted to the 5G App and Service Catalogue, in order to be on-boarded, by using the resource Id already created.
5. The Catalogue notifies to Sender that the onboarding request has been received.
6. The Catalogue performs different actions on the NSD:
   a. Validates the integrity and authenticity of the NSD by using the security info provided by it.
   b. Checks that all the VNF Packages used by the NSD are present in the catalogue.
   c. Checks that all mandatory elements are included.
d. Translates the NSD from the generic format into the specific format expected at the underlying NFV Orchestrators.

7. Once the NSD is properly translated in the needed specific formats, the Catalogue requests the Onboarding of the translated NSDs into the underlaying MANO catalogues.

8. The NFVO notifies to the Catalogue that the onboarding request have been received.

9. The NFVO notifies to the Catalogue that the NSD onboarding operation has been completed.

Once all the NSD onboarding requests are completed, the Catalogue notifies to the Sender the operation result.

4.5.6. NS Instantiation

The NS instantiation workflow is shown in Figure 55. When the instantiation of a NS is requested, the service provider has to specify the NFVI environment where the NS will be deployed. The Service Orchestrator checks the NSDs of the NS and validates the existence of the referenced VNFs in the 5G-MEDIA Catalogue. In the next step, the SO checks also the
availability of resources in the NFVI by requesting the corresponding VIM and proceeds to the necessary reservations to instantiate the VNFs. After that, network connectivity allocation is considered and VNF instantiation is triggered for every VNF of the NS. The LFC of the instantiated VNFs is passed to the corresponding VNFMs. As a last step, the VNFFG is established by linking VNFs with the set connectivity network and the service provided is informed about the successful instantiation of the NS.

Figure 55 – Network Service instantiation

4.5.7. NS scale out running FaaS

Figure 56 shows the MANO scale-out message flow. OpenWhisk does not offer action resizing API out of the box, even though Docker containers can be resized on the fly. While we believe that Step 5 does not make much sense for short lived actions, we can support it by providing a specialized OpenWhisk action that will communicate resizing requests to a resizing service. Step 5 is useful for long running containers off-loaded to Container Orchestrator Engine by the FaaS framework. Step 6 amounts to regular action invocations. Scale-out mechanism can be used for automatically spawning FaaS based VNFs on-demand, where the autoscaling group is treated as a group with cardinality [0, N] rather than [1, N].
4.5.8. VNF forwarding graph optimization in runtime

Figure 57 shows the work flow of Service Orchestration, including its subcomponents - VNF placement and VNFFG manager - without the cognitive network optimizer. In such a case, the service provider contacts the VNF placement component to instantiate VNFs to form the NS. It will then trigger the instantiation of VNFFGs by the help of the VNF placement component which is responsible to create VNFFG instances via VNFM/VIM. When a user requests a new session, the VNFFG request will be sent to the VNFFG manager. If all VNFs are up and running and there are sufficient resources available, the ingress IP address of the VNFFG instance will be returned to the user. Otherwise, the VNFFG manager sends a request to the VNF management component to instantiate new VNF instances.
The case where the Cognitive Network Optimizer is also integrated into the workflow is shown in Figure 58. The service provider creates a policy to trigger the optimization process running in the background which periodically optimizes the resource usage to satisfy system performance. The other scenario that can trigger the optimization process is when new data is collected. The Machine Learning Engine will process this data to predict user demand and resource availability. It will trigger the Policy/Optimization component to instantiate or remove VNFs. User requests can happen at any time in this period. The way of handling user requests is presented in the previous diagram.
4.5.9. NS monitoring in MAPE loop

Figure 59 shows the work flow to collect metrics and monitor NSs. After infrastructure owner or service provider defines the group of metrics that should be collected from the NFVI or the application environment (i.e. by configuring the corresponding telemetry tools), a special service, called “exporter”, is responsible to periodically collect these metrics and publish them into the Kafka bus. These (raw) metrics are stored in the database and also are adapted/combined together to compose NSs monitoring objects to be consumed by the Cognitive Network Optimizer, as well any other service or component of the SDK/SVP is interested in.
4.5.10. Catalogue authentication and authorization

Each request to the Catalogue is intercepted by a specific Authentication/Authorization filter that verifies the presence and the validity of the token in the request, extracts the logged user information from such token, verifies the authorization needed to access the requested resource and grants access in case, relying on the Authentication/Authorization/Accounting (AAA) service.

The access token is provided through a dedicated API to support not interactive clients such as the SDK CLI.

The Figure 60 shows the high level working flow of the authentication and authorization on the Catalogue when the SDK invokes its API. An insight view of the AAA components will be provided in the deliverable D4.1 “5G-MEDIA Catalogue APIs and Network Apps”\textsuperscript{92}.

![Figure 60 – Catalogue authentication/authorization](image)

4.5.11. Accounting

The accounting service collects the information coming from the monitoring to track the NS/VNF instantiation for each user/tenant in the 5G-MEDIA, and the resource consumption of the VDU on the NFVI (typically cpu cycles, memory and disk usage), with support to other more relevant media specific resources (such as bandwidth). The main goal is to support billing in commercial and non-commercial business models that will be identified later on in the project. The Figure 61 shows the working flow of the accounting service from the perspective of the monitoring.

\textsuperscript{92} http://www.5gmedia.eu/outcomes/deliverables/
Figure 61 – Accounting service
5. 5G-MEDIA functions and services portfolio

5.1. ETSI VNF package specifications

ETSI VNF Package is considered as a standard and uniform way for VNF providers to deliver VNFs. This specification takes TOSCA YAML Cloud Service ARchive (CSAR) as a good basis for VNF packaging, however, this kind of files has some constraints when talking about security.

A CSAR file is, mainly, a .zip file with a well-defined structure. As specified before, the security of the file needs to be improved. For this reason, the whole VNF package, or the manifest file inside it, is digitally signed. Moreover, each of the artefacts contained on the package could be signed. In addition, asymmetric or symmetric encryption may be added.

The main structure of a VNF package is:

- **Metadata information**: information about the name of the package, its version, its provider, the TOSCA and CSAR versions, etc. This information is included in a TOSCA.meta file which defines that descriptions are included in the YAML file (VNFD). However, instead of having this metadata file, the information that contains could be included directly on the YAML file.

- **VNF Descriptor**: A VNF Descriptor is the main TOSCA definitions YAML file in which metadata for package onboarding and VNF management is presented. This means that it specifies the VNF properties and requirements. Most important of them are:
  - Resources needed by the VNF like the amount and type of Virtual Compute, the storage needed or the required networking.
  - Connectivity specifications, such as:
    - External and Internal Connection Points, which are described on Connection Points Descriptors (CPD).
    - Internal Virtual Links, which are described on VLDs.
  - Lifecycle management behavior, operations and configuration.
  - References to software images, lifecycle management scripts and other files contained on the VNF package. It could also have references to other files referred by the package.
  - Affinity and other policy rules.
  - Deployment flavours (size-bounded deployment configurations).

- **Manifest file**: this file provides package integrity and authenticity. It contains some metadata about the VNF (name, provider, version, release date), and it also contains the URL for each artifact contained on the VNF package. In order to add security to the package, a hash code could be added to each artifact specifying the algorithm used to generate this hash code. The manifest file also contains naming conventions for name-value pairs. Finally, the manifest file could be signed by the provider for higher security. An example of a manifest file with high security is shown in Figure 6.2.
Software images: this software images are needed to run the VNF.

Change History file: this file is a human readable text file where all changes made on VNF package are versioned, tracked and inventoried.

Testing files directory: the goal of this kind of files is to enable the validation of the VNF package. The provider includes other files on this directory, containing more necessary information, such as the description of the tests.

Licensing information directory: this directory includes a license term for the whole VNF as well as other license terms for other artifacts included on the package if their license is different from the VNF one.

Certificate files: files with extension.cert which aim to add security to the VNF package. With these certificate files, the VNF provider can digitally sign some of the artifacts inside the VNF package.

Some optional additional files: these files are intended to manage the VNF, they could be scripts, vendor-specific files, etc.

The detailed technical specifications for VNF and NS descriptors as well the packaging model implemented in 5G-MEDIA project are provided in D4.1: 5G-MEDIA Catalogue APIs and Network Apps.

5.2. List of VNFs in 5G-MEDIA project

Table 7 summarizes the VNFs which are implemented in the scope of 5G-MEDIA project, as well the UCs where each one is utilized. In Annex I, a detailed description is provided for each enlisted VNF.
<table>
<thead>
<tr>
<th>VNF</th>
<th>Short Description</th>
<th>UC1</th>
<th>UC2</th>
<th>UC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3DMediaTranscoder</td>
<td>Near real time transcoding of 3D Time Varying Meshes to varying levels of quality.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>vBuffer</td>
<td>A VNF able to buffer 3D content of the last X seconds of the 3D TI session.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>vReplay</td>
<td>A VNF that when triggered saves the contents of vBuffer to a persistent storage medium.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>vCompression Engine</td>
<td>Performs compression and encoding/decoding of A/V material</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Media Process Engine</td>
<td>Performs signal switching of A/V signals, needs buffering/frame store and some kind of timing information/clock for signal synchronisation</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Speech-to-Text Engine (part of the Cognitive Services)</td>
<td>Recognition of the A/V material’s audio signal which will be decoded into text.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Image/Face Recognition (part of the Cognitive Services)</td>
<td>Detection of objects within the A/V material with a context-aware text-based output.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Mid and Edge Cache - vCache</td>
<td>Cache server where the user is connected to, can be deployed according to a hierarchy of mid/edge caches orchestrated/managed by the vCDN, in order to serve the media content near to the end user, reducing latency and offering a better QoS/QoE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Edge Transcoding Unit - vTranscoder</td>
<td>Transcoding unit for media content, transcodes the media content according to the</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### VNFs and Short Description

<table>
<thead>
<tr>
<th>VNF</th>
<th>Short Description</th>
<th>UC1</th>
<th>UC2</th>
<th>UC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS/QoE offered to the end user, can be centralized or deployed at the edge for reducing latency and improving QoS/QoE</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Transmitting VoD from media libraries according to user profiling and preferences and live streaming from external sources. Can be eventually replicated and serves as the root for vCache/vTranscoder hierarchies</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Collect, organize and share media content to be offered on-demand to authorized end users</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Offers different streams related to the various viewing angles and/or different audio tracks</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Cache metadata content over a small to medium distributed caching system</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Apart from the above listed VNFs, 5G-MEDIA also makes use of the PNFs shown in Table 8.

### PNFs and Short Description

<table>
<thead>
<tr>
<th>PNF</th>
<th>Short Description</th>
<th>UC1</th>
<th>UC2</th>
<th>UC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Server</td>
<td>This PNF acts as the central authority responsible for synchronising the game state among players.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5G-MEDIA Gateway</td>
<td>SDI-to-IP/IP-to-SDI Gateway that encapsulates video, audio (and metadata) for transfer and constitutes the link between the venue/the broadcasting centre and the edge cloud.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
A more detailed description of implemented VNFs can be found in Annex I and also in “D2.2: 5G-MEDIA Requirements and Use Case Refinement”.
6. Conclusions

The 5G-MEDIA project will rely on background open source technologies and Phase 1 project results to implement the 5G-MEDIA platform. In this deliverable, we have presented an overview of the assessment of existing solutions and explained the selection of specific components.

In particular, one key decision during the architecture design has been the selection of the MANO framework for the realization of the SVP. The 5G-MEDIA partners have considered four candidate solutions (OSM v3.0, OPNFV, OpenBaton, SONATA rel. v3) and analysed their advantages and disadvantages with respect to relevant criteria for the 5G-MEDIA project. An additional hands-on assessment of SONATA and OSM (considered as the most prominent candidates) has been done to provide supplementary feedback on the evaluation. The outcome of this process showed that there is no clear better solution between these two candidates. Yet, since OSM is supported by a large community (including TID, one of the telecom operators of the project) is a more promising option for the sustainability plans of the project. Considering these aspects, the 5G-MEDIA partners have decided to use OSM to realize the NFV-MANO functionalities of the platform.

Upon this decision, an initial high-level architecture was designed to depict the main platform components. With respect to the SDK, the 5G-MEDIA project relies on the SONATA-SDK emulator that is already integrated to OSM and aims to extend its functionalities by enabling the emulation of FaaS scenarios. In addition, the 5G-MEDIA SDK will build tools to allow defining Media Service forwarding graphs, including FaaS VNFs. 5G-MEDIA will also integrate and extend the NS/VNFs Catalogues and Repositories of SELFNET to build the 5G-MEDIA catalogue and the CogNet MAPE infrastructure and data flow to realize Media Service MAPE component.

The main technical developments of the 5G-MEDIA project focus on: i) the implementation of the MEDIA Service MAPE component, the implementation of two new VIMs (FaaS VIM, OpenNebula VIM); ii) the implementation of the SDK tools and the adaptation of the emulator to support FaaS; iii) the extension of the SELFNET catalogue with respect to the latest standards Catalogue to provide a NFV MANO platform-agnostic in terms of formats and syntax for Network Service (NS) descriptors and VNF Package information model and iv) the implementation of new media VNFs.

The initial architectural design presented in this deliverable provides a common reference point for the implementation activities of the project. The specifications of each platform component will be further elaborated in WP3, WP4 and WP5 deliverables, which supplement the documentation on the design of the system.
7. Annex I – Potentially exploitable resources of 5G PPP phase 1 projects

7.1. The project COHERENT

7.1.1. Overview

The project aims to design, develop and showcase a novel control framework for 5G heterogeneous radio networks, which leverages the proper abstraction of Physical and Medium Access Control (MAC) layers in the network. It introduces a novel programmable control framework to offer operators a powerful means to dynamically and efficiently control wireless network resources, and thus significantly improve capacity, spectrum reuse efficiency, energy efficiency and user experience in their increasing complex Heterogeneous Mobile Networks (HMN).

The main concept proposed by the project is illustrated in Figure 63. The key idea is to develop an additional programmable control framework, on the top of current control planes of operators' mobile networks, being aware of underlying network topology, radio environment and traffic conditions, and being able to efficiently coordinate wireless network resources cross the border of cells. The centralized network view with sufficient but abstracted information on radio links, interference, network topology, load information and physical layer reality (e.g. the use of Coordinated Multipoint (CoMP), or Device to Device (D2D) links, moving cells/relays management) is essential to enable optimal resource allocation in the network.

COHERENT develops a programmable control framework, which utilizes the centralized network view, represented by different types of network graphs, to efficiently coordinate resource allocation in HMN. To this end, the project carefully evaluates and separates control functions which must be placed at local and which will benefit from the centralized implementation. It provides several abstractions for the control functions and develops an SDK to support customized control programs for different resource allocation schemes in different layers/parts of HMN. With the programmable control approach, the automation in network-wide resource allocation is significantly improved and operators have the ability for radio network innovations.
### 7.1.2. Technical approach

In COHERENT, network graphs representing the low-layer reality are formed at the new control layer by extracting parameters from Physical and MAC layers of Radio Access Technologies (RATs). These graphs are network abstraction because they are constructed by partial information of network and thus, capture only certain level of states in the network. The example of network graphs formed by a network is shown in Figure 64. These abstracted network graphs provide sufficient details about network topology, connectivity and interference. It is the place to implement low-level coordination algorithms to deal with coordination between User Equipment (UEs) and individual cells. The network graphs are separated in regions so that each region has no connection to node in other regions. This is the way to reduce the graph size and thus the computation complexity.

Low layer network graphs are processed by the network virtualization layer, so they are further abstracted to provide basic network topology information. For instance, a high-level abstracted network graph may only include base stations but leave out UEs. With the abstract network model, the high-level coordination can be done regarding resource allocation at the cell level, for instance, the load balancing and spectrum management among cells.

![Illustration of COHERENT concept](image-url)
The developed Service Development Kit (SDK) enables the programmability in the control framework. To offer operators the ability to apply policies on radio resource provisioning and coordinate resources according to service requirements, interfaces have been developed to link control programs with operators’ Operations Administration and Maintenance (OAM) system. A special application server extracts all the necessary data from OAM’s databases and creates the network graph by invoking the appropriate algorithm. Control programs are also run in the application server. The controller is responsible to obtain the output of control programs and do corresponding configuration to the underlying radio networks.

7.1.3. Innovation

COHERENT makes a concrete step beyond the state of the art by introducing a novel control framework to offer advanced inter-cell coordination in HMN. The need for efficient control and coordination in HMN is strong due to the densification of and spectrum sharing in the network. Challenges are how to efficiently coordinate the resource in network wide with accepted signalling overhead, as well as the flexibility of control to meet different service requirements and operators’ operation strategies. COHERENT provides an additional control layer on top of existing control plane in current mobile networks. It introduces a new network abstraction to reduce the information needed for network-wide control, and the programmability to control functions for open and flexible control and coordination implementations for HMN.

7.2. The project CHARISMA

7.2.1. Overview

CHARISMA proposes the use of Ethernet in the fronthaul as the means to provide a common technology platform to achieve low-latency and security provision, while benefiting from cost reductions, performance monitoring and networking techniques. The approach will permit optimization of the wireless and supporting optical distribution network performance. In
particular, CHARISMA examines D2D communications and virtualized cloud-to-cloud communications by bringing together researchers from several domains/sub-domains. For this reason, the work plan broadly groups the project activities in two axes: (i) definition/specification and implementation of specific security use cases (horizontal activities), and (ii) development of the Secure Open Access Architecture and Physical layer security technologies (vertical activities). The general purpose of this separation is to allow, through the vertical activities, the development of the integral solutions (architecture-level security design, open access platform design modules, physical security techniques and cross-layer protocols) in support of network flexibility and their integration to practical environments.

The protection of public and private services accessed over open access networks is a central new feature of CHARISMA. The development of technologies ensuring enhanced security that can be natively supported in 5G networks has an important impact on the quality and the number of services that can be supported and be enjoyed by the public. In general, more security will increase the number of satisfied end-users of mobile services, and thereby the revenue of operators. New services will potentially emerge, such as trustworthy mobile banking on 5G devices and the native support for low latency services e.g. for cooperative base stations and for the Internet of things (IoT). Furthermore, additional opportunities to provide secure exchange of information and sensitive data over the wireless channel, besides using SIM cards or shared keys, will enable users to become more efficient in situations that necessitate secure distributed or remote decision making. An example of a potential application of this research is secure e-governance that can enhance participation in public affairs without compromising anonymity. Besides both, fixed and mobile users covering almost all parts of the society, the sectors of telecommunication industries as well as network operators in the EU will greatly benefit from the successful endeavours of this research, increasing their global competitiveness. These industries face new challenges as the number of users increases; secrecy approaches that can adapt to variable data rates without compromising security are essential for designing new successful products and services. Two well-known use cases that can benefit from the CHARISMA developments are:

- Virtual Open Access Network Operator (Service Provider)
- Distributed Security enabling IoT

Both use cases are thoroughly investigated in CHARISMA and act as the key drivers of the two corresponding pilot demonstrators of the project.

7.2.2. Technical approach

Figure 65 addresses the challenge of providing holistic end-to-end security for high QoS (low-latency) services in a shared and converged advanced 5G network environment. Currently, security is treated as a component embedded into the corresponding management system as a monolithic system unable to dynamically adapt to new, open future infrastructure requirements. One aspect of providing dynamic security requires that service providers rely on other network infrastructure providers, who may not be trustworthy. In addition, as CHARISMA allows users to select service providers, this offers a further security advantage. Our holistic approach allows network providers to: (i) offer security at the physical level; (ii)
authenticate and authorize infrastructures; and (iii) allows the community to customize the security they require.

![Diagram](image)

*Figure 65 – CHARISMA virtualised C&M converged 5G wireless/wireline infrastructure scenario*

At the ICT systems physical level, CHARISMA employs two different complementary security mechanisms using physical layer security (PLS) techniques and encryption at the physical layer (PHY encryption). First, PLS technologies can guarantee secrecy by 1) symmetric key encryption using keys generated through shared randomness techniques, 2) information theoretic encoding schemes in scenarios with vanishing probability of secrecy outage, e.g. femtocell settings and wireless smart home, 3) optical layer encryption. At the interface of heterogeneous networks, CHARISMA proposes to use a combination of PLS and PHY encryption methods in order to control the rate information flow in the two directions.

Through mixed PLS and PHY encryption, CHARISMA will: a) secure transmissions in an unmanaged decentralized network where other methods are impractical; b) control password generation at the physical level according to security requirements; c) oversee communication channel impairments; d) provide provable security; and e) control the secrecy level in physical layer transmissions by triaging security, throughput and energy consumption.

The Open Access model implies a single infrastructure provider (InP) serving several services providers, with physical infrastructure shared through the control and Management (C&M) systems, which becomes a fundamental enabler to provide required flexibility, elasticity, and programmability required for 5G access core networks. CHARISMA follows a role-based access model approach to authentication and authorization to secure the different relationships between InP, NP and SPs, leveraging from the Resource-Owner Role-Actor model of FP7
GEYSERS project\textsuperscript{93} by implementing recursive resource access delegation between the InP and the SPs (SAML, XACML). Security virtualization at the SP level will be adopted, to externalize security from the C&M system. Consequently, run-time verification will be ensured. Extending the C&M planes to include virtualization and security, the dynamic management approach adapts to different “security” levels. The security requirements of network virtualisation environments identify a set of potential attacks and an adaptive set of security mechanisms can then be implemented. The following figure depicts the security service value-chain of the CHARISMA concept.

CHARISMA is designed to resolve contention, as in C-RAN upstream traffic, by allocating content caching features to maximise utilisation efficiency. Cloud-RAN design is therefore of critical importance to minimise network latency and power consumption. We use Flow Engine technology in the Trust Node router IP core to provide a hierarchical solution to traffic management. Here, the lowest router hierarchy contains controllers, whilst the home router is the next highest level connecting personal computers etc. The Trust Node router technology will be used as second level router, employing the IPv6-based hierarchical 6Tree routing concept. With 2 bytes selection code, 56 bits of the 64-bit IPv6 network address prefix remain for 6Tree addressing. The Trust Node ignores the second part of the IPv6 address but accepts the IEEE 48-bit MAC identifier (RFC4291). This Trust Node approach also allows user-controlled QoS, such that service bandwidth is a user preference. Mediated by the YouQoS server, this provides the control-place interface between operators and end-users. Subscribers state their preferences or select a predefined profile and the user or his/her devices signal flow information to the YouQoS server. The YouQoS Server implements algorithms to map user preferences and flow information to optimum QoS parameters, which are sent to the Classifier and Marker Unit (CMU) instance in the 5G network infrastructure. The CMU annotates the packet priority to each packet and forwards them to the scheduler unit. CMY realization is via an SDN OpenFlow switch. Current low-end OpenFlow switching is capable of serving at least 1,000 subscribers.

7.2.3. Innovation

The key innovations related to the CHARISMA project are:

- Security challenges in a virtualized network environment and at the hardware-device level, which includes: a) secret key generation in non-reciprocal channels (i.e. optical/wireless), b) security in multi-layer data streaming, and c) multi-user wireless access security
- Low-latency of end-to-end connectivity across a heterogeneous, multi-domain access infrastructure.

\textsuperscript{93} http://www.geyser-project.eu/
7.3. The project SESAME

7.3.1. Overview

As a common practice, a mobile network operator may operate the Radio Access Network, the mobile core networks (e.g., Evolved Packet Core (EPC) for LTE), as well as Data Centres for providing Cloud services, and thus, normally possesses full control of its network infrastructure. SESAME consolidates the concept of delivering small cell coverage coupled with a virtualised execution platform, according to the ‘as a Service’ fashion, in order to allow multi-tenant operation. Under this point of view, the provider of SESAME services, that could be for example the party that owns the Cloud-Enabled Small Cell (CESC) infrastructure, will be in the role of the hypervisor (to continue the virtualisation metaphor) and possess view of the bare metal – the underlying real network and its performance – from which the virtual network management views will be derived.

Virtualisation in small cells drives also the inclusion of mobile-edge computing capabilities that allow acceleration of content, services and applications, increasing responsiveness from the edge. Mobile-edge computing consists of cloud servers running at the edge of a mobile network and performing specific tasks that could not be achieved with traditional network infrastructure. This allows enriching mobile subscriber’s experience and, at the same time, operators can open the radio network edge to third-party partners, allowing them to rapidly deploy innovative applications and services. Figure 66 demonstrates this concept. Therefore, an important aspect of current approach is that a service provider may exploit SESAME’s architecture to compose and operate on an end-to-end basis his own ‘virtual’ network, provided by different mobile operators and infrastructure owners on top of a set of diverse physical infrastructure. Another instance could be a third party (as e.g., a virtual mobile network operator, a data centre provider or another business entity) that would act as end-to-end provider, without actually owning any physical infrastructure. That part would sign agreements with physical infrastructure owners, for instance municipalities, big stadiums, etc., in the areas it wishes to provide network services without deploying infrastructure, in order to cover sporadic large-scale events for example. The same would be the case for contracting data centre operators in strategic locations to complete a full network solution.

Figure 66 – SESAME impact on service deployment
### 7.3.2. Technical approach

The key innovation proposed by this project focuses not only on providing multi-operator radio access capacity with virtualised small cells (that can be integrated within the tenant’s operator infrastructures), but also on evolving the small cell concept to embrace a virtualised execution environment for delivering cloud services at network’s edge. The software architecture of SESAME with emphasis on the functional elements and interfaces is presented in Figure 67. The thorough examination that follows, describes the work split in the proof-of-concept design and implementation.

As shown in Figure 67, SESAME architecture allows multiple network operators (tenants) to provide services to their users through a set of CESC s deployed, owned and managed by a third party (i.e. the CESC provider). In this way, operators can extend the capacity of their own 4G/5G RAN infrastructures (represented in the figure in the form of a macrocell) within specific areas where the deployment of their own infrastructure could be expensive and/or inefficient, as it would be the case of e.g., highly dense areas where massive numbers of Small Cells would be needed to provide the expected services. In addition to capacity extension, CESC s are equipped with a virtualised execution environment, materialised in the form of the Light DC that allows the provision of mobile-edge computing capabilities to the mobile operators for enhancing the user experience and the agility in the service delivery. The Light DC will also support the execution of VNFs for carrying out the virtualisation of the Small Cell access. In this regard, network functions supporting MOCN features are expected to be executed within the Light DC. Backhaul and fronthaul transmission resources will be also part of CESC, allowing for the required connectivity.
The management functionalities necessary for the service provisioning and operation of the CESC platform are also depicted in Figure 67 within the CESC Manager (CESCM) environment. It includes different components. The NFVO is in charge of realising NSs on the virtualised infrastructure and exposes interfaces to interact with the OSS/BSS system of the CESC Provider for high level service management (e.g., exchange of network service descriptors and SLAs for each tenant). The NFVO coordinates groups of VNF instances that jointly realise a more complex function. To that end, the NFVO uses the services exposed by the VNF Manager, which will be in charge of the instantiation, update, query, scaling and termination of the VNFs.

The CESCM includes also a set of supporting tools for coordinating the operation of the CESCs, such as the monitoring and analytics to capture relevant indicators of the network operation, and the CESC Radio Access Manager to configure and optimise the operation of radio settings. Self-x functionalities are also enabled by the CESC platform within the multi-operator environment. In addition, the Radio Access Manager supports capacity planning functions for CESC provider to be able to anticipate demand fluctuation patterns and provision resources accordingly. Another central function within the CESC management is the Element Management System (EMS) virtualisation. This function is responsible for partitioning the single whole-cell management view into multiple virtual-cell management views, one per operator/tenant. In this way, a virtualised small cell with a set of (limited) management functionalities can be made visible to the Network Management System (NMS) of the operator through the management interface in order to e.g., collect performance counters, configure neighbour lists for a proper mobility management, etc.

The CESC management functions are built upon the services provided by VIM for appropriately managing, monitoring and optimising the overall operation of Light DC resources. The role of VIM is essential for the deployment of NFV services and to form and provide a layer of NFV resources to be made available to the CESCM functions. The NFV resources will be ultimately offered as a set of APIs that will allow the execution of network services over the decentralised small cells located at the edge of the network.

7.3.3. Innovation

SESAME develops a novel system architecture based on the concept of CESC that will be a clear step toward the implementation of 5G systems, promoting a denser cellular network and accelerating the deployment of Small Cell base stations. SESAME will devise a novel architecture for 5G networks combining edge computing and global network optimisation and will promote the concept of multi-tenancy for different operators using a third-party infrastructure, thus promoting new market opportunities. As building blocks, SESAME will decouple RAN and computational power, relying on a non-x86 Light DC and it will develop a flexible infrastructure management framework.

Additionally, SESAME not only embraces the idea of a denser 5G network deploying low cost, low power, high performance computing platforms deploying small cell Base Stations but it fosters the whole concept of CESC.

SESAME architecture fully relies on the cloud computing paradigm to leverage a system in which networking and computing capabilities are provided as services. In particular, SESAME will rely on different network elements such as Light DCs, the VIM and the CESCM. Their
combination is meant to accomplish the mission of enabling a cloud computing network of small cells.

SESAME adopts the SDN approach for decoupling control from forwarding functions and to develop solutions that involve SDNs in which the nodes' functionality can be programmatically modified. This will be an essential step to provision resources, accelerate network deployment, circumvent interference issues and bring services nearer to the end-users, and it will allow further management and optimisation policies, allowing more flexible control and solutions for resource allocation. Finally, the SDN approach will be used in SESAME to foster multi-tenancy, make an open to new market players and enable network flexibility.

The following technical elements will be considered in the CESC development in SESAME.

- Small Cell
- Self-x features
- Light DC
- Hardware accelerator support
- Backhaul/fronthaul resources
- CESC Management (CESCM)
  - NFV Orchestrator (NFVO)
  - VNF Manager
  - EMS Virtualisation
  - Catalogues
  - Radio Access Manager
8. Annex II – List of 5G-MEDIA virtual network functions

8.1. Application layer functions

- **vTranscoder**: This application-specific network function will be responsible for transcoding 3D-appearance data of each player to varying levels of quality in near real-time manner to support adaptive streaming of TVMs to spectators and or game clients.

- **3D Media QoE**: This virtual network function will be responsible in measuring application-layer QoS/QoE metrics related to the immersive media application from all players and spectators and provide suggestions to the MANO layer for orchestrating other VNFs at the application layer and potentially invoking network layer functionality such as QoS prioritisation and traffic steering functions, to improve the QoE of players and spectators.

- **vReplay**: The vReplay network function will be instantiated by the application layer when a game specific highlight event occurs. The purpose of the network function will be to save the contents of the vBuffer node to a persistent storage media making the content available for offline on-demand viewing by spectators. This VNF, similar to vBuffer, can be designed in a generic manner to also support other use cases since its functionality is not directly dependent on the media application’s specific data format. vReplay is amenable to the FaaS approach. The game events stream from which replay clips are being created is inconsistent. Thus, it does not make sense economically to allocate resources of the platform to execute vReplay continuously. Rather, vReplay should be instantaneously elastic, highly reactive, and event driven. This will allow the game provider to save on operational costs considerably and therefore compete more efficiently while preserving the required QoS to the gamers.

- **vCompression Engine**: Compression and Encoding/Decoding of A/V material. It requires high computational power and buffering and needs to support different codecs and formats.

- **Media Process Engine**: Signal switching of A/V signals. It requires high computational power, buffering/frame store and timing information/clock for signal synchronisation. As the MPE placement is after the vCompression Engine, audio and video signals must be switched/mixed in a format that allows to do that. MPEG2/H264/H265 IBP frame compression schema cannot be used since those streams are not editable. So, when an MPE is needed a JPEG2000 schema seems to be appropriate. Another option could be the use of MPEG2/H264/H265 with only Intra frame compression if these streams allow cut editing. The MPE can have two roles: serving as a video switcher and outputting auxiliary signals as a broadcast router.
  
  **Video switcher**: Signals coming from vCompression Engine in the edge feed the MPE. One of these signals is to be switched/mixed towards the output by means of the commands coming from the TV centre. Such commands belong to the control plane.
  
  **Broadcast router**: Other VNFs such as Cognitive Services need a pre-selected signal as an input. MPE routes such signals as a broadcast router whereas the signals are controlled by the TV centre through the control plane.
• **Cognitive Services**: Allows for (automatic) enrichment of video content with additional information such as speech-to-text and face recognition.

  *Speech-to-text Engine*: Recognition of the A/V material’s audio signal which will be decoded into text. It is based on deep learning techniques thus part of the Cognitive Services which enable next generation media enrichment.

  *Image/Face Recognition Engine*: Detection of objects within the A/V material with a context-aware text-based output. It is based on deep learning techniques thus part of the Cognitive Services, which enable next generation media enrichment.

• **UHD Streaming server**: The origin server within the vCDN domain, acts as the root server for the vCaches/vTranscoders hierarchies and it’s composed by the following functions: Transcoding unit User preference and profiling Media Catalogue access

• **Media library**: Function to collect, organise and share media content to be requested on-demand by the authorised end users.

• **Content personalisation server**: It offers different streams related to the various viewing angles and/or different audio tracks.

• **Stats collector**: Component in charge of monitoring specific application level parameters in all the nodes in the vCaches/vTranscoders hierarchies. This collected information is used by the Cognitive Network Optimiser to adapt the service delivery on the base of users’ requests and vCaches/vTranscoders health status.

• **Application layer traffic steering** (to the best vCache, vTranscoder or origin server): Function in charge of balancing the incoming users’ requests between the vCaches and vTranscoders in the vCDN domain and determine which vCache/vTranscoder fits better the incoming user request for a media service, also in terms of proximity to the end user. The configuration of the traffic steering for the application layer is delegated to the Cognitive Network Optimiser, which will process application layer and infrastructure layer service monitoring stats to optimise the traffic flows instantiated in the 5G network.

• **Distributed cache**: Service to provide small to medium distributed cache content from one producer to multiple consumers based on the packaging as a VNF of NetFlix open source project “Hollow”\(^6\). The most common use case is about the dissemination of “metadata”\(^5\) avoidig the centralized approach that could lead to latency and bandwidth limitations, as long as reliability issues. As declared in the official web site, the size of the dataset to be cached should be in terms of GB: “A good rule of thumb in 2017: KB, MB, and often GB, but not TB or PB” \(^6\).

---

\(^5\) NetFlix Hollow: [https://github.com/Netflix/hollow](https://github.com/Netflix/hollow)

\(^6\) Typical use case for NetFlix hollow: [https://medium.com/netflix-techblog/netflixoss-announcing-hollow-5f710eefca4b](https://medium.com/netflix-techblog/netflixoss-announcing-hollow-5f710eefca4b)

\(^6\) Typical dataset size for NetFlix hollow: [https://hollow.how/](https://hollow.how/)
8.2. Generic media functions

- **vBuffer**: The vBuffer virtual function is a generic media function that will be responsible to memory-buffer the last $X$ seconds of game-state and 3D-appearance data. This buffered content will then be able to support the execution of other VNFS like vReplay. While in this use case the vBuffer node will be used to save game-state and 3D-appearance data specific to this use case, the design of the VNF could be generic to support any content that needs to be memory-buffered eventually supporting other use cases as well.

- **Edge Transcoding unit**: Transcoding unit deployed close to the end user with the aims of reducing latency and offer a better QoS/QoE.

- **Edge Cache**: Caches deployed close to the end users with the aim of reducing latency and offer a better QoS/QoE.

8.3. Network layer functions

- **QoS Prioritisation**: The QoS prioritisation capability controls the way traffic is forwarded over the underlying network paths. Prioritisation can be accomplished by several mechanisms, including expedited forwarding techniques through various queuing disciplines and through bandwidth reservation for high priority traffic. Prioritisation will be required when the existing network path is unable to deliver the network performance the application requires, for example to reduce latency or jitter on congested links.

- **Traffic Steering**: Traffic steering is concerned with routing traffic over paths that meet the performance objectives of the application stream. In cases where the default path with QoS prioritisation enabled is insufficient a new path may be required. Techniques for achieving this include the use of multi-topology routing at the network level, source routing or the use of an SDN controller-enabled network layer where paths are determined dynamically for specific flows.

- **Multicast**: This network function will be responsible to deliver the same application content to many recipients (like spectators and game clients) in a network-efficient way by eliminating duplicate transmissions that would otherwise be needed for unicast traffic.

- **Traffic classification**: function to inspect traffic and trigger appropriate traffic routing decisions at the application level. Data collected and aggregated by proper QoS/QoE monitor functions will be consumed by the Cognitive Optimiser to select the best path along which the service will be delivered.

- **Security Functions**: function to provide basic network security like firewall protection of systems and services as well as user authentication to permit only authorised access to the services, functions, platform and network.

- **Frontend/Backend security functions**: for instance, vFirewall, vDDoS, vIPS, to protect users’ data.

- **Traffic Steering and QoS Prioritisation functions**: making use of queue management techniques and path selection, including the use of SDN controllers.
• **QoS Monitor**: function to collect and aggregate information/statistics from the network layer. The resulted data set can be consumed by the Cognitive Network Optimiser for the service chain optimisation.