Deliverable D6.3

Management for the Plug & Play Control Plane

Editor:	Pietro G. Giardina, Nextworks
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual delivery date:	31/05/2019
Actual delivery date:	06/06/2019
Suggested readers:	Network Administrators, Integrators, DevOps (Development and Operations) Professionals, Telecommunication Operators, Service Providers
Version:	1.0
Total number of pages:	59
Keywords:	Network Slicing, Plug & Play, Slice Management, Serverless

Abstract

This document reports the design and prototype implementation of the SliceNet Plug & Play Manager, as the lifecycle coordinator of the Plug & Play control features deployed within the SliceNet cognitive management platform. The evolution of the Plug & Play approach is largely described in the document, to present how it has been enhanced towards the support of truly programmable and per-slice customized control and management logics within the SliceNet management framework. Following a cloud-native approach leveraging on microservices architectures, containerized applications and serverless functions, the Plug & Play Manager aims at making the whole SliceNet cognitive management platform more and more flexible and ready to cope with highly dynamic and heterogeneous requirements imposed by 5G technologies, services and vertical industry. From a software prototype perspective, the Plug & Play manager allows to dynamically deploy per-slice applications within the SliceNet cognitive management platform for slice tailored control and management purposes as containerized applications in Kubernetes clusters and as serverless applications in OpenWhisk environments.

Disclaimer

The research leading to these results has received funding from the European Union Horizon 2020 Programme under grant agreement number H2020-ICT-2014-2/761913.

Impressum

[Full project title] End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks

[Short project title] SliceNet

[Number and title of work-package] WP6 - 5G Multi-Domain Slice Management Plane

[Number and title of task] T6.3 - Management for the Plug & Play Control Plane

[Document title] Management for the Plug & Play Control Plane

[Editor] Pietro G. Giardina, Nextworks

[Work-package leader:] Thuy Truong, DELL EMC

Copyright notice

Executive summary

SliceNet defines and implements a cognitive management platform for 5G networks that aims at delivering vertical-tailored and Quality of Experience (QoE) driven network slices and services. SliceNet fulfils heterogeneous vertical requirements by means of highly programmable network slice control and management logics. As one of its core principles, SliceNet implements a Plug & Play control framework that was originally defined to enable high degree of vertical driven customization of end-to-end slice runtime control, with tailored exposure of control primitives towards verticals.

With the aim of achieving a highly programmable and on-demand slice control and management architecture, while going beyond current practices based on monolithic and all-in-one frameworks that supervise and control from a single point a various and heterogeneous plethora of services, technologies, resource domains, vendor solutions, SliceNet has revised and enhanced its Plug & Play paradigm. The new goal is to combine in the SliceNet system architecture the traditional cross-slice and cross-service management functions with more programmable, cloud-native oriented and granular per-slice logics that can help in achieving and addressing heterogeneous requirements imposed by verticals and Digital Service Providers. In practice, the original Plug & Play approach, mostly oriented towards customization of control exposure towards the verticals, is now enhanced for having per-slice differentiation of control and management logics within the SliceNet cognitive management platform itself.

In this context, the Plug & Play Manager becomes the main enabler and lifecycle coordinator for all those per-slice control and management applications that can be containerized, deployed and configured to be part of the SliceNet cognitive management platform and therefore achieve a customized and fine-granular control and management architecture. The Plug & Play Manager leverages on microservices architectures and container orchestrators to manage, beyond the original Plug & Play control instances, SliceNet control plane services, QoE optimization applications and data analytics functions as per-slice containerized applications to be deployed within the SliceNet platform according to slice requirements and capabilities required by verticals and Digital Service Providers. In addition, as a further cloud-native evolution of the SliceNet system architecture, serverless applications are also integrated as part of the new Plug & Play approach to run short-lived cognition applications (e.g. for QoE monitoring or data analytics) following the Function as a Service paradigm. All of these are integrated in the Plug & Play Manager design, that provide common and unified lifecycle management of these per-slice control and management applications, hiding the complexity of their deployment and configuration.

The software prototype of the Plug & Play Manager is released as open source and is able to manage the deployment and configuration of per-slice control and management containerized applications in Kubernetes clusters, as well as of serverless applications in OpenWhisk Function as a Service platforms. A single Plug & Play Manager software prototype therefore applies to both deployment models of the SliceNet system architecture, and can be deployed at both the Digital Service Provider and Network Service Provider levels.

List of authors

Institution	Author
NXW	Pietro G. Giardina
NXW	Giacomo Bernini
NXW	Ali Nejabati
TEI	Ciriaco Angelo
TEI	Giacomo Borlizzi
CSE	Konstantinos Koutsopoulos
DELL	Thuy Truong

Deliverable Reviewers

Internal Reviewer	Institution
Salvatore Spadaro	UPC
Qi Wang	UWS

Table of Contents

List of tables

List of figures

Abbreviations

API	Application Programming Interface
BBU	Base Band Unit
CN	Core Network
CP	Control Plane
CSP	Communication Service Providers
DSC	Digital Service Customer
DSP	Digital Service Provider
E2E	End-to-End
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
ID	Identifier
LCM	Lifecycle Management
MANO	Management and Orchestration
ME	Mobile Edge
MEC	Mobile/Multi-access Edge Computing
NF	Network Function
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NSP	Network Service Provider
NST	Network Slice Template
OAM	Operations, administration and maintenance
OSS	Operations Support System
P&P	Plug & Play
PNF	Physical network Function
PoP	Point-of-Presence
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
SDN	Software Defined Networks
SDO	Standards Developing Organization
SLA	Service Level Agreement
SliceNet	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks
SS-O	Slice Service Orchestrator

UC	Use Case
UE	User Equipment
VM	Virtual Machine
WG	Working Group
WP	Work Package

1 Introduction

1.1 Motivation and scope clarification

SliceNet defines and implements a cognitive management platform for 5G networks that aims at delivering vertical-tailored and Quality of Experience (QoE) driven network slices and services. SliceNet fulfils a wide set of heterogeneous vertical requirements by means of highly programmable network slice control and management logics. As one of its core principles, SliceNet implements a Plug & Play control framework that was originally conceived to enable verticals having direct runtime control of their end-to-end slices on top of a tailored slice view.

More than that, this Plug & Play control approach has been evolved towards a more comprehensive programmable and on-demand slice control and management paradigm that goes beyond the runtime slice control exposed to verticals. The SliceNet Plug & Play in its enhanced definition embraces all the per-slice control and management applications that can be deployed to customize and specialize the slice management logics.

Therefore, the SliceNet Plug & Play features include not only the planned runtime tailored control instances and functions offered to verticals, but also those additional per-slice control and management applications that reside within the SliceNet management platform and enable customized and vertical-oriented slice management. This approach applies to both Network Service Provider (NSP) and Digital Service Provider (DSP) levels, and makes the whole SliceNet management platform even more programmable and agile. For each given slice, a set of dedicated control and management functions are exposed to verticals as a way to offer them tailored slice runtime control, while other per-slice applications are deployed at runtime to support the programmable operation and maintenance of 5G network slices and services from DSP and NSP perspectives.

This evolved of Plug & Play architecture highly leverages on the original Plug & Play microservices and cloud-native approach, and enables the evolution of current monolithic management architectures towards specific per-slice and per-purpose management logics, in line with current trends in the telecom industry for highly programmable and dynamic network and service management frameworks in support of 5G networks and slices. In this context, the Plug & Play management functionalities allow to manage the lifecycle of these vertical-oriented and internal SliceNet platform per-slice control and management functions as containerized applications through a common and unified approach. In particular, as shown in Figure 1, the Plug & Play Manager is positioned in the orchestration sub-plane within the SliceNet system architecture, and under the coordination of the Service and Slice Orchestrator (SS-O) it enables the deployment of several per-slice control and management applications, including:

i. Plug & Play control instances, exposed to verticals to offer them customized runtime control primitives and tailored end-to-end slice views
ii. SliceNet Control Plane Services, as part of the SliceNet NSP level platform and providing slice-level runtime control primitives for internal use in the cognitive control loops
iii. End-to-end QoE monitor and data analytics applications, as part of the per-slice customized and vertical-oriented cognitive features within the SliceNet DSP level management platform
iv. End-to-end QoE optimization applications, to apply vertical and slice specific end-to-end optimization logics as part of the cognitive control loop within the SliceNet DSP level management platform

1.2 Objectives of the document

This deliverable aims to achieve a set of technical objectives related with the management of the Plug & Play control plane in SliceNet:

- Evolve and enhance the SliceNet Plug & Play principles and paradigm to make the SliceNet cognitive management platform more and more programmable and agile towards the support of 5G services and network slices
- Design an extensible and common Plug & Play management architecture to enable
- Define interfaces and APIs of the Plug & Play manager to ease its integration within the SliceNet management platform
- Provide a Plug & Play manager software prototype to coordinate the lifecycle of those per-slice control and management applications that are deployed on-demand within the SliceNet cognitive management platform

1.3 Document structure

This document is organized as follows:

- Section 2 provides a brief overview concerning the current trend in managing the lifecycle of slice management and control functions, focusing on solutions implemented in 5GPPP Phase 2 and Phase 3 projects, as well as analysing the with a gap analysis towards Plug & Play approach. Moreover, a brief insight on how other 5GPPP Phase 2 projects deal with verticals and slice exposure is included;
- Section 3 defines the main principles and objectives for the SliceNet Plug & Play management, highlighting the new extended scope of the Plug & Play domain with respect to the original project view;
- Section 4 is the core part of the document and provides a full description of the Plug & Play manager architecture, translating the principles and objectives into concrete functions and features. Details about lifecycle management workflows, information model and APIs are also provided;
- Section 5 provides a description of the Plug & Play manager software prototype that has been implemented following the architecture and APIs defined in section 4. Here, a description of a preliminary integration and validation of the Plug & Play manager prototype in the context of the SmartGrid use case is also provided;
- Section 5.1 provides some conclusions for the work reported and highlights on future work.

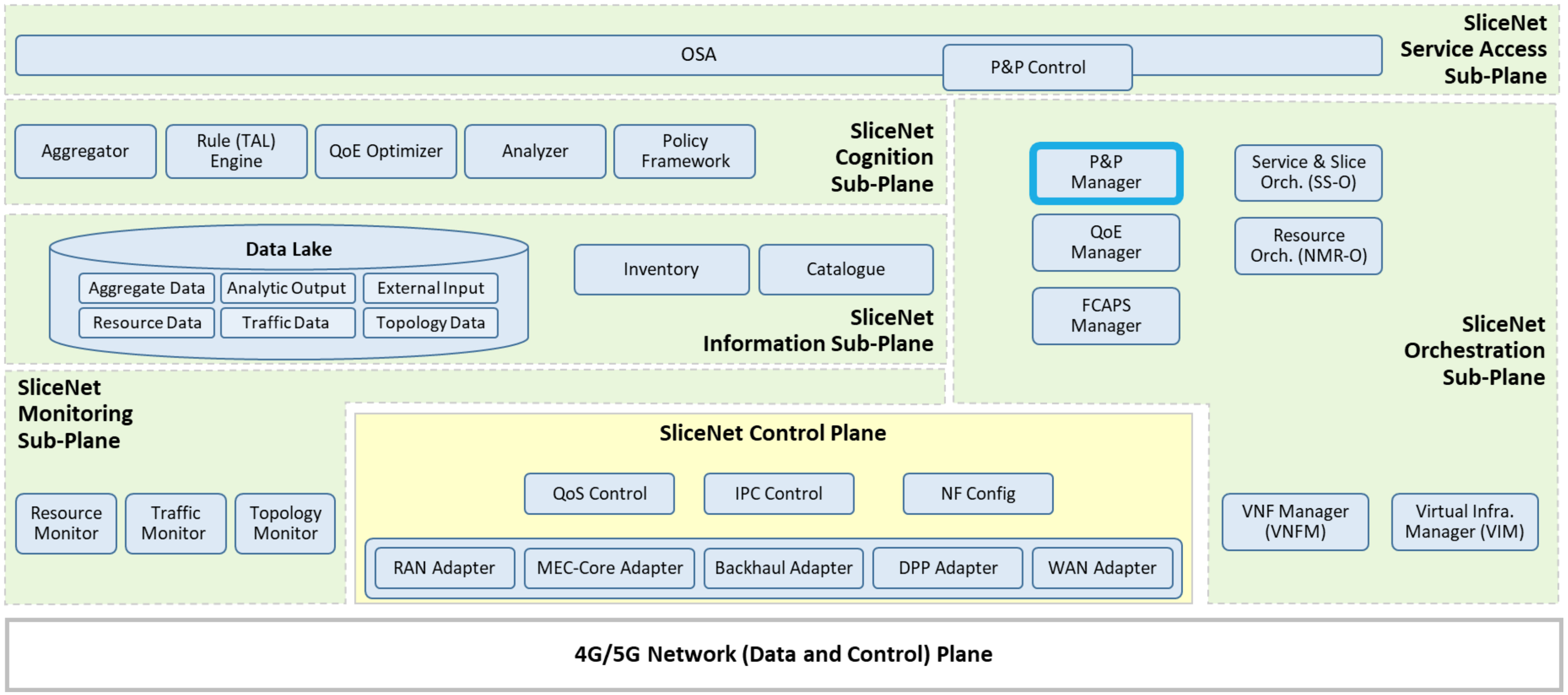


Figure 1 Plug & Play management within the SliceNet system architecture

2 Lifecycle Management of Slice Control and Management Functions

Traditionally, the network management systems used by network operators and service providers to deploy services for their customers are based on monolithic and all-in-one frameworks that supervise and control from a single point a various and heterogeneous plethora of technologies, resource domains, vendor solutions. Recently, this approach is being overcame by the adoption of more programmable and softwarized solutions, leveraging on the NFV and SDN software networks principles, which enable abstraction and unified control and management procedures and APIs with the aim of avoiding any vendor lock-in for network operators and service providers.

However, this software-based transformation still has to be completed in operational networks and environments. And 5G is already challenging the NFV and SDN paradigms and software networks management solutions as it opens to new vertical oriented requirements on the one hand, and to a completely new mobile network architecture and functional decomposition. In addition, network slicing put more and more requirements in terms of programmability, flexibility and performances for the control and management of vertical slices with very divergent needs in terms of QoS.

This requires a paradigm shift in how the network and cloud infrastructures (both physical and virtualized), services and network slices are managed and controlled to follow the dynamicity and highest granularity of verticals and end user services. In particular, the transition from current all-in-one control and management frameworks towards more granular (and possibly distributed) per-slice functions would allow to ease the fulfilment of diverse vertical requirements by relying on simpler control and management logics for the control and management of a subset of infrastructure resources and network domains, services, among those at disposal and deployed by network operators and service providers.

This vision matches the SliceNet view for the lifecycle management of slice control and management functions. Indeed, besides its original scope defined and described in deliverables D2.3 [1], D2.4 [2] and D4.1 [3], the SliceNet Plug & Play (P&P) principle is extended to also cover the customization of per-slice control and management functions to be deployed on-demand as part of the Network Service Providers (NSPs) and Digital Service Providers (DSPs) management frameworks. Therefore, while in D2.3, D2.4 and D4.1 the P&P features were oriented to expose a tailored slice control to verticals, now we apply a similar approach for those per-slice control and management functions that are not exposed to verticals but are integrated in the NSPs and DSPs management frameworks to implement more granular, scalable and isolated logics at the slice level.

The following sub-sections briefly first introduce how network slice control and management is currently tackled by existing (or under development) solutions in the state-of-the-art, and then describe the SliceNet vision and solution as enabled by the P&P approach.

2.1 Current trends and approaches in the state-of-the-art

The brief survey on current existing solutions for slice control and management reported in this section is split in three different main categories: i) solutions adopted in other 5GPPP Phase 2 and Phase 3 projects, ii) solutions adopted by open source frameworks, iii) solutions adopted by relevant telecommunication industry players.

2.1.1 5GPPP Phase 2 and Phase 3 R&D Projects

A thorough analysis of relevant R&D projects solutions with respect to the SliceNet P&P approach was performed in deliverable D4.1. While in D4.1 the focus of this analysis was on how these projects were tackling runtime control exposure of the provisioned services and slices, some more considerations are now required for what concern their approaches in control and management of slices within their respective management platforms. The target R&D projects analysed in D4.1 are still relevant, and are: 5G-Transformer, MATILDA, 5GCity, 5G ESSENCE, 5G MEDIA.

5G-Transformer [4] aims to transform today's rigid mobile transport networks into an SDN/NFV-based Mobile Transport and Computing Platform (5GT-MTP), that brings the network slicing paradigm into mobile transport networks by provisioning and managing slices tailored to the specific needs of vertical industries. 5G-Transformer follows an approach for network slicing that is strictly bound to the NFV principles and architecture, and defines two main components for slice and services management and orchestration: the Vertical Slicer and the Service Orchestrator. In particular, the control and management of slices is delegated to the Vertical Slicer component, which is a monolithic slice orchestration tool embedding all the logics for any kind vertical slice and requirement. Therefore, no dedicated per-slice control and management functions are considered as part of the 5G-Transformer overall management solution.

MATILDA [5] provides a novel holistic 5G end-to-end services operational framework tackling the overall lifecycle of design, development and orchestration of 5G-ready applications and 5G network services over programmable infrastructure, following a unified programmability model and a set of control abstractions. Similar to 5G-Transformer, MATILDA implements a 5G network service orchestration framework that mostly follows the NFV architecture and functional split, with a high degree of customization offered to verticals and end users at the design phase. The control and management of slices, mapped to 5G network services, is applied by means of legacy monolithic NFV MANO components (NFV Orchestrator, VNF Managers) with a low degree of per-slice customization within the MATILDA platform itself in terms of dedicated runtime functions.

5GCity [6] designs, develops and deploys a distributed cloud and radio platform for municipalities and infrastructure owners who can act as 5G Neutral Hosts. The main goal of 5GCity is to build and deploy a common, multi-tenant, open platform that extends the centralized cloud model to the extreme edge of the network. In particular 5GCity offers a smart city slice management platform to municipalities to enable them to offer different smart city services following the 5G Neutral Host approach. The management platform is again fully aligned with the ETSI NFV MANO architecture, augmented with Multi-Access Edge Computing (MEC) capabilities to offload latency critical applications at the edge of the network. As in the other R&D projects above, the 5GCity management platform does not foresee any per-slice control and management function to be operated at runtime, and is therefore deployed as single monolithic set of NFV MANO oriented components.

5G ESSENCE [7] provides a highly flexible and scalable platform to provide a Neutral Host market for Small Cell as a Service products, taking advantage of edge cloud computing and Cloud-RAN solutions. While the 5G ESSENCE management platform is again aligned with the ETSI NFV MANO architecture, the implementation of the Small Cell as a Service approach requires the deployment of dedicated per slice Radio Resource Management functions and controllers that are partially exposed to verticals. Therefore, 5G ESSENCE employs in its platform a mix of traditional monolithic network and service management components, and more dynamic and per-slice dedicated runtime functions, even if covering the Radio Access Network (RAN) domain only.

5G MEDIA [8] aims at innovating media-related applications by investigating how these applications and the underlying 5G network should be coupled and interwork to the benefit of both. The 5G-MEDIA framework leverages on SDN and NFV principles and technologies for offering an integrated Service Virtualization Platform (SVP) capable of handling the lifecycle management and monitoring of media services and applications in 5G networks. Therefore 5G-MEDIA delivers an integrated programmable platform solution for the design, development and operations of media applications by providing mechanisms to flexibly adapt services to network changing conditions and react upon them. This is implemented in compliance and full alignment with the ETSI NFV MANO architecture, which is integrated with cloud-native frameworks for the implementation of media functions as serverless functions. However, this serverless high-level of flexibility and per-service dynamicity is kept at the application level only, while the 5G-MEDIA platform itself does not support any per-slice control and management runtime function.

In addition to these 5GPPP Phase 2 projects, other three Phase 3 projects are relevant to this analysis in terms of their solutions for control and management functions for end-to-end slices spanning multiple domains and sites in support of vertical trials and experiments: i) 5G EVE, ii) 5G-VINNI, iii) 5Genesis.

5G EVE [9] is building a truly integrated end-to-end 5G experimentation facility, where the validation services offered by four sites (France, Greece, Italy, Spain) are exposed to verticals and their applications as a unified platform service. An interworking framework has the role of abstracting the specific logics and tools available in each site, exposing the whole multi-site infrastructure through a unified and site-agnostic information model towards the upper layers of the 5G EVE architecture. The whole 5G EVE architecture, especially the internals of each site facility and the interworking framework, is highly leveraging on the ETSI NFV MANO architecture and principles for the provisioning and execution of the vertical experiments, following a hierarchical approach at the interworking level to coordinate the heterogeneous management and orchestration tools at each site. At the time of writing, 5G EVE does not foresee any per-slice or per-experiment control and management function that is meant to be operated at runtime at either the level of the site facilities, the interworking framework or the integrated portal.

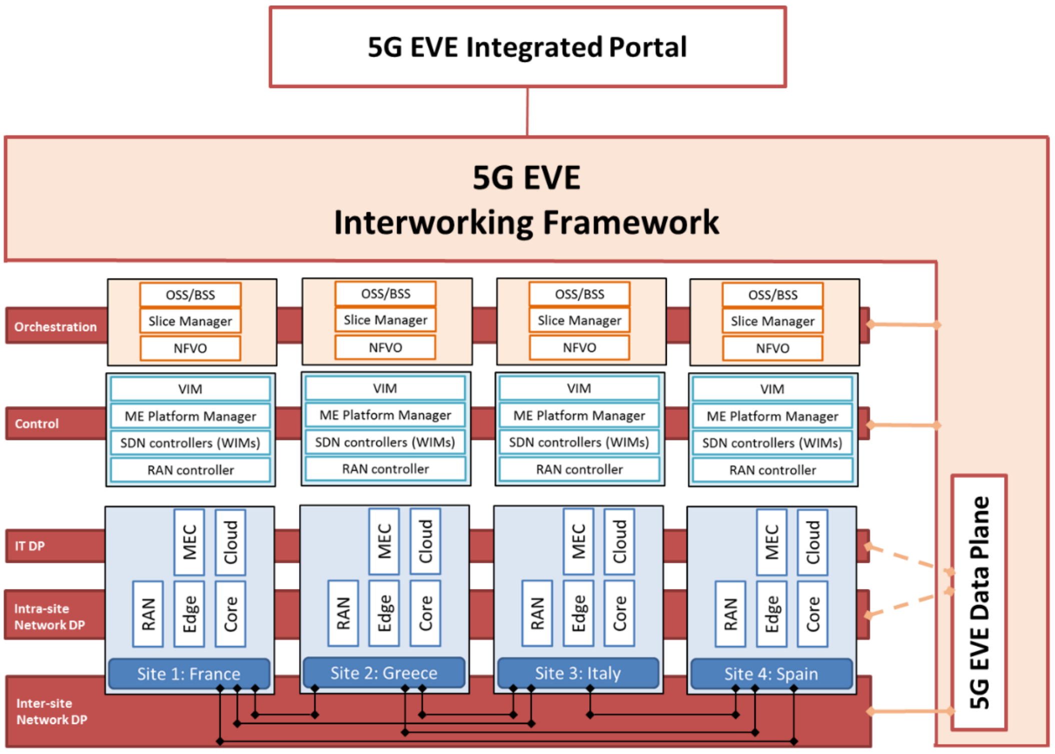


Figure 2 5G EVE multi-site interworking approach (source 5G EVE D3.1 [10])

5G-VINNI [11] is developing an end-to-end 5G facility that can be used to first demonstrate the practical implementation of infrastructure to support the key 5G KPIs, and then to allow vertical industries to test and validate specific applications that are dependent upon those KPIs. In its management and orchestration architecture, 5G-VINNI leverages on the ETSI NFV MANO architecture and procedures to apply a recursive approach to manage the lifecycle of services and resources at two different levels: i) at each facility in the 5G-VINNI infrastructure, ii) at the end-to-end level towards the vertical customers to enable their experiments. At the time of writing there is no evidence of the intention in 5G-VINNI to follow any per-slice approach in the implementation of slice control and management functions on top of its ETSI NFV MANO based management and orchestration.

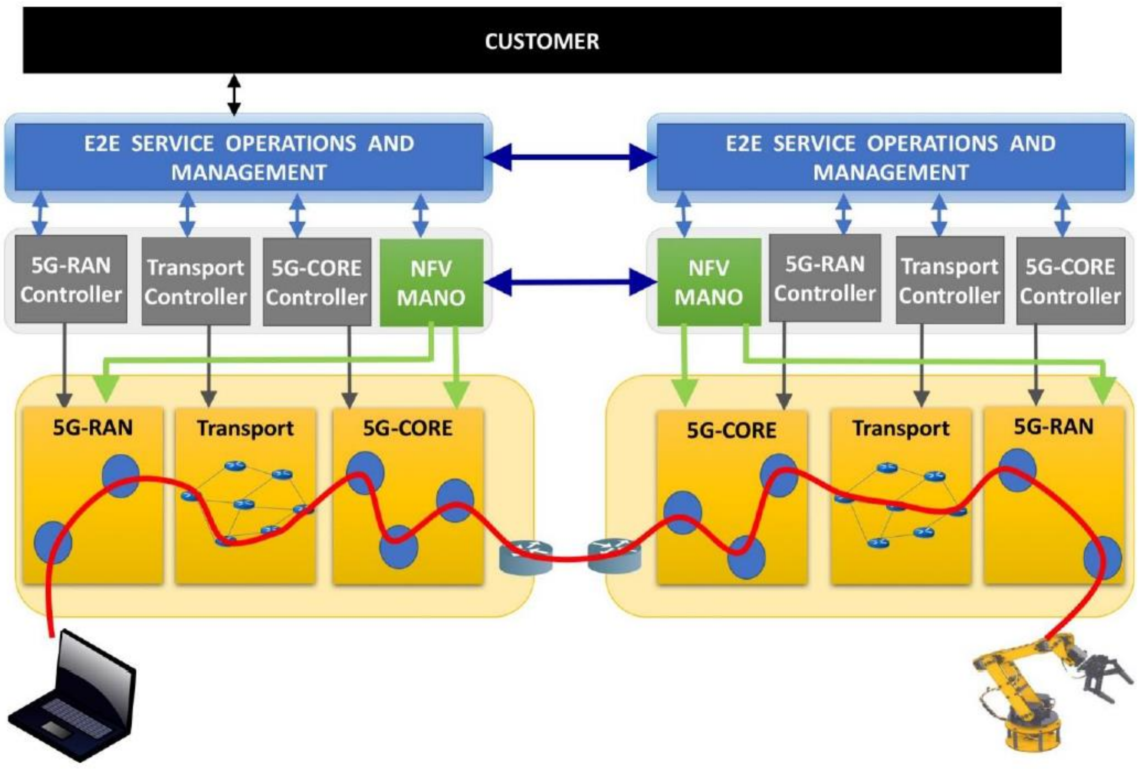


Figure 3 5G-VINNI end-to-end Management and Orchestration (source: 5G-VINNI D1.1 [12])

5Genesis [13] is designing and developing and end-to-end facility to support 5G experimentation for the validation of 5G capabilities and 5G KPIs for different services coming from various vertical industries. This is performed over an integrated infrastructure composed by five experimentation platforms in Athens, Berlin, Malaga, Limassol and Surrey, providing common access framework for verticals, alignment with relevant 5G standards and multi-domain end-to-end orchestration. The 5Genesis reference architecture, as shown Figure 4, follows a classical layered and modular approach, and its management and orchestration layer is aligned with ETSI NFV MANO functionalities and architecture decomposition. In particular, network slice and NFV management features are defined as monolithic components, with no evidence of per-slice dedicated runtime control and management functions [14].

In summary, most of these relevant 5GPPP Phase 2 and Phase 3 projects still lack of supporting a truly dynamic and per-slice programmable lifecycle of slice control and management functions. This is mostly due to the fact that all of them are aligned (in few cases fully) with the ETSI NFV MANO architecture and procedures for the management and orchestration of network slices. However, as it is actually the case of SliceNet (as described in section 2.2), this should not have prevented having a combination and integration of traditional monolithic orchestration and management functions with per-slice functions to be activated, configured and operated independently for each vertical slice.

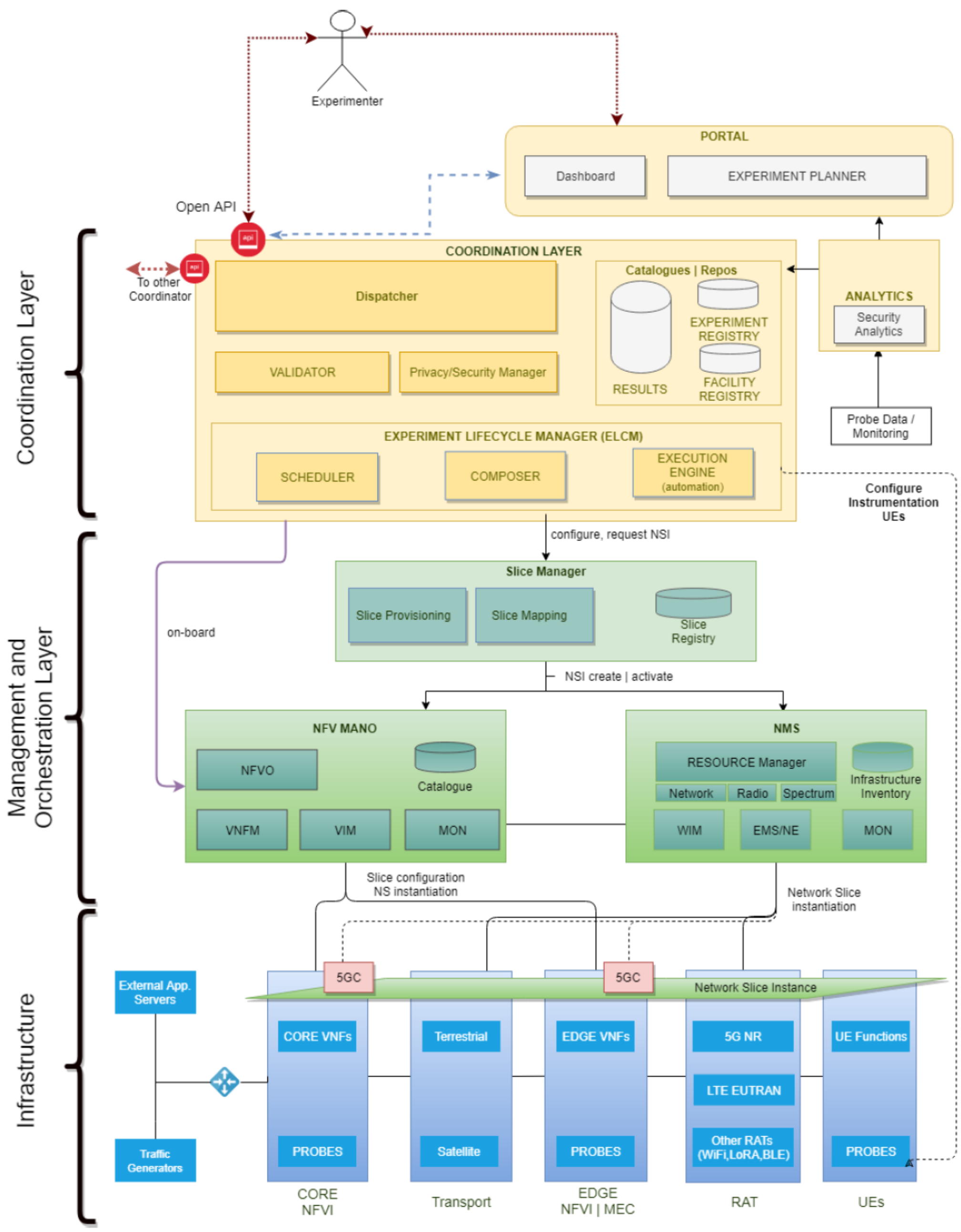


Figure 4 5Genesis overall architecture (source: 5Geneisis D2.2 [14])

2.1.2 NFV and network slice management open source frameworks

Most of the R&D projects mentioned in the previous section make use of existing open source frameworks for what concern the implementation of their NFV and network slice management platforms. Similarly, the use of open source projects and tools at large is the preferred strategy for many European network operators and vendors as a cheaper and faster way to implement proof-of-concepts for 5G services and network slicing. Two of the most relevant, widely used and de-facto standard NFV and network slice management open source frameworks are certainly Open source MANO (OSM) and ONAP.

OSM [15] is an operator-led ETSI community that is delivering a production-quality open source MANO stack aligned with ETSI NFV Information Models and targets to meet the requirements of production NFV networks. OSM is widely used in the European research community (including R&D projects), and aims at providing the ETSI NFV MANO reference implementation. Currently, in its Release 5 [16], it supports ETSI NFV standard northbound REST APIs and provides slice management functionalities based on 3GPP specifications and models.

ONAP [17] is being developed under the umbrella of the Linux Foundation, is mainly supported by AT&T, and aims at providing a global and massive scale (multi-site and multi virtualized infrastructure) orchestration capabilities for both physical and virtual network elements. ONAP provides a common set of northbound REST APIs that are open and interoperable, and by supporting YANG and TOSCA data models it simplifies integration with multiple Virtualized Infrastructure Managers, VNF Managers, SDN Controllers, and even legacy equipment. As a main difference with respect to OSM, it does not yet support ETSI NFV REST APIs and data models, while it is closer to service definitions from TM Forum. In the latest upcoming Release 4 Dublin [18], ONAP also provides initial support for network slice management and orchestration.

Both OSM and ONAP are evolving in the direction of implementing highly dynamic, scalable and per-purpose (i.e. per-service, per-slice, per-function) control and management logics. They can be already installed and deployed as decoupled microservices running in containers, thus looking towards cloud-native deployments and environments. Moreover, it is clear in their development roadmap the target of becoming ready for managing and orchestrating 5G services and slices in a truly programmable way evolving the monolithic (ETSI NFV) MANO approaches.

2.1.3 Trend in the telecommunication industry

The telco industry is more and more approaching cloud-native solutions to meet the extremely dynamic and agile capabilities of 5G networks. The high degree of programmability 5G networks will bring are imposing telecom operators and service providers to investigate on new control and management architectures able to fulfil and benefit of such a flexible paradigm. Cloud-native architectures can help to enhance traditional network and service control and management monolithic systems, as they radically reduce the costs of building and operating systems at scale. Moreover, they dramatically shorten time to market and accelerate the innovation by allowing developing, testing, deploying, and independently scaling stateless service components with well-defined functionality rather than stateful monolithic, multi-functional services. As 5G networks start taking shape of a commercial reality, it becomes increasingly clear that the future of 5G networks is going to be cloud-native [19][20].

In the telco domain, applications are created for data, control and management planes processing. Due to critical processing requirements, applications deployed in the telecom clouds should be resilient, offer ultra-high performance, very low latency, extreme scalability and be real-time or close to real time to fulfil the end-user and vertical requirements. Therefore, with 5G, these data, control and management plane applications should be stateless (which makes them fault tolerant and scalable), employ the microservices architecture (minimal configuration at each component with minimal dependencies), capitalise on containerisation technology (faster, lighter), easy to orchestrate automatically via intent based policies, and last but still of key relevance, fully embrace mature open source software. All of these are features common to any cloud native service.

In line with these considerations, some of the key telco players are already evolving their solutions and products towards cloud-native architectures. It is the case of Ericsson, which is looking at cloud-native telecom applications as the key solution for its 5G Core set of products, from data to control and lifecycle management tools [21]. Following this approach, Ericsson is looking at agility, flexibility and programmability offered by cloud-native software platforms based on microservice architecture for easing the automated orchestration and co-existence of its 5G Cloud Core solution with 5G New Radio (NR) Stand-Alone (SA) and Non-Standalone (NSA), 4G, 3G and 2G accesses technologies.

On the telecom standardization side, several Standard Development Organizations (SDOs) are also looking to new programmable paradigms for the definition of next generation network control and management architectures. As a relevant example, the ETSI Zero touch and Service Management

(ZSM) Industry Specification Group (ISG) [22] was created early in 2018 by major telco industry players around the world with the objective of looking at cloud-based network infrastructure and functions and leverage on cloud-native principles to define a new way of network and service management and orchestration able to meet the challenges introduced by 5G. Full automation in network management and operation is currently targeted by ETSI ZSM in its architectural work item ZSM-002 [23] that is defining the Zero-Touch Network and Service Management architecture leveraging on a Service Based Architecture (SBA) design where loosely coupled pluggable intra and inter domain management services and applications interoperate through an integration fabric that facilitate their communication offering service registration and discovery features. This ETSI ZSM approach enables to implement network and service management applications as microservices, inline with the cloud-native principles.

2.2 The SliceNet vision

The complexity of 5G network infrastructures and services require extremely agile and programmable management, control and orchestration functions and logics in order to deliver slices to verticals with proper QoS and QoE requirements. This means that per-slice, per-service and per-purpose control and management functions are needed within NSPs and DSPs management platforms to decouple the logics (in terms of actions, decisions, analysis) to be applied for heterogeneous slices with very diverse objective functions in terms of performances and Service Level Agreements.

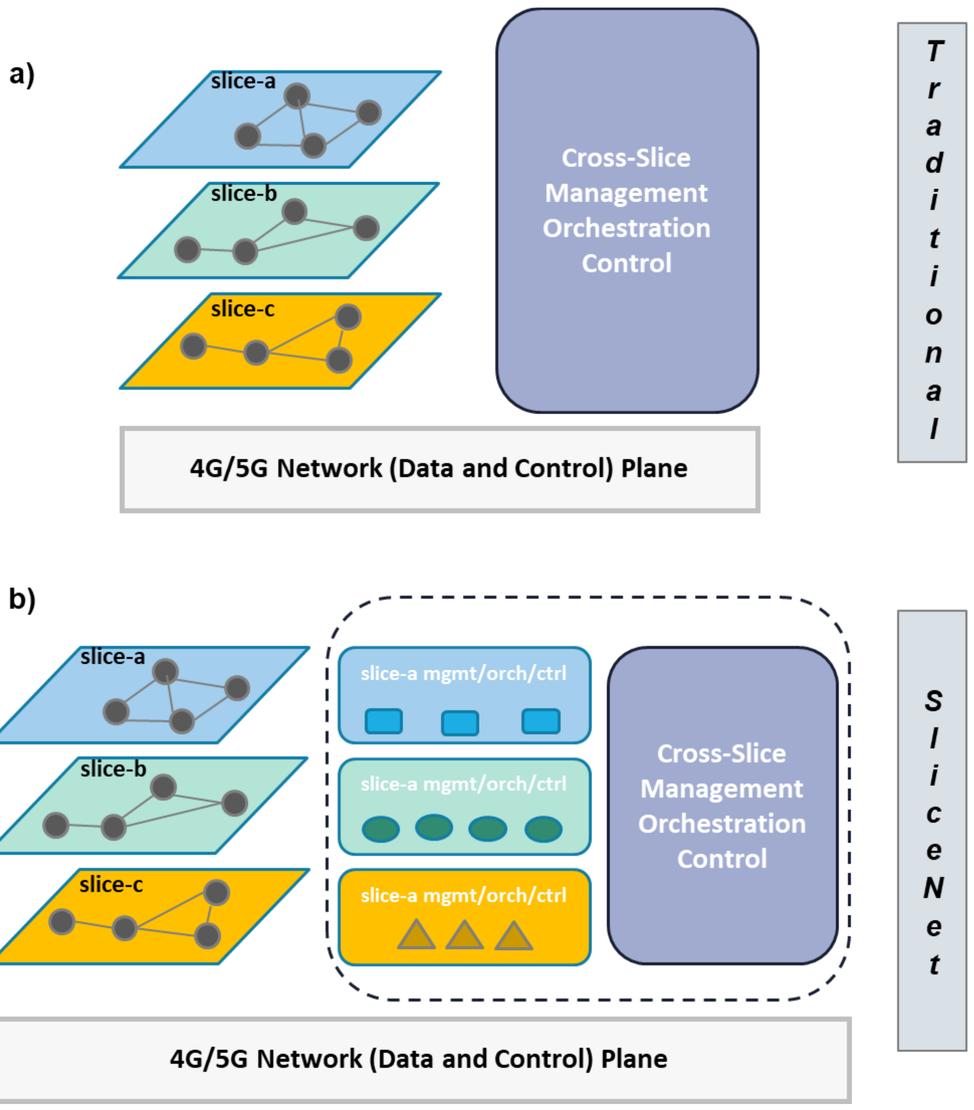

a)
slice-a
slice-b
slice-c
Cross-Slice Management Orchestration Control
Traditional
4G/5G Network (Data and Control) Plane
b)
slice-a
slice-b
slice-c
slice-a mgmt/orch/ctrl
slice-a mgmt/orch/ctrl
slice-a mgmt/orch/ctrl
Cross-Slice Management Orchestration Control
SliceNet
4G/5G Network (Data and Control) Plane

Figure 5 Traditional monolithic management approach (a) vs. SliceNet approach (b)

SliceNet follows this direction by implementing a management platform where traditional monolithic cross-slice management and orchestration functions are combined with per-slice and per-service functions that are activated, configured and operated within the NSP and DSP management platforms only when needed and with the aim of fulfilling the specific needs of (and agreements with) verticals and customers. The main enabler of this innovative approach is the SliceNet P&P solution, which is now evolved and enhanced with respect to what originally designed in deliverables D2.3, D2.4 and D4.1. In particular, the P&P Manager is still the lifecycle manager of the P&P control functions deployed and exposed to verticals to offer them tailored runtime control of their slices and services, and in addition it becomes responsible for the ad-hoc deployment of additional P&P control and management functions that have internal scope to the NSP and DSP (thus are not exposed) and that allow to distribute the complexity of slice management over multiple atomic and decoupled functions. Moreover, this extended scope of the SliceNet P&P allows to implement (at least part of) the slice control and management functions as decoupled microservices running in containers, as shown in Figure 5, going in the direction of a cloud-native management paradigm, where distributed and scalable management logics are applied.

This evolution applies at both NSP and DSP levels, as detailed in the next chapter, and allows to have per-slice and per-service functions to be dynamically used for both runtime control (i.e. related to the SliceNet control plane) and cognition purposes (i.e. related to the SliceNet cognition sub-plane). For what concerns the slice cognition capabilities and functions in particular, this evolved P&P approach allows to easily implement and support within the NSPs and DSPs platforms diverse cognition schemes (in terms of machine learning algorithms, data analysis mechanisms, etc.) for different verticals and slices. In addition, the plug of new machine learning algorithms (or in general new per-slice control and management functions), becomes easier as it does not disrupt the lifecycle of other slices.

3 Plug & Play Manager: Principles and Objectives

SliceNet provides a cognitive management framework for end-to-end network slices, aiming to deliver truly customized and tailored services to verticals. This translates into the need of a highly flexible, dynamic and agile management and orchestration approach, capable to adapt and fine-tune its slice and service control and management logics to heterogeneous vertical requirements. Following this direction, the SliceNet P&P approach was initially targeting the customization of slice control exposure towards verticals, through an innovative combination of tailored control functions, APIs and tools to be offered to verticals to let them plug their own control logics and specialize their slice instances according to their needs.

On top of that, this P&P principle has been extended towards a per-slice customization of the SliceNet control and management functionalities. This aims at achieving higher flexibility, dynamicity and agility within the SliceNet cognitive management framework for providing vertical-tailored and per-slice control and management functions within the SliceNet platform. In this context, the P&P Manager represents the lifecycle orchestrator of this combined customization of vertical-tailored slice control exposure, and internal per-slice control and management logics.

Following the mapping between the SliceNet reference architecture introduced in section 1 and the business roles (i.e. NSP, DSP and vertical actors) defined in D2.2 [24], the P&P Manager fits in both the NSP and DSP flavours of the SliceNet cognitive management framework. Figure 6 shows the SliceNet system architecture at the NSP level; here, the P&P manager is part of the orchestration sub-plane in the NSP management plane (that includes the cognition, information and monitoring planes as well) and takes care of the lifecycle of the SliceNet control plane services. As defined in D4.3 [25], the SliceNet control plane services provide per-slice runtime control logics and allow fulfilling cognition-driven closure of the control loop by exposing common APIs to hide the complexity of the underlying 5G infrastructure from a management perspective. In particular, the QoS control, Inter-PoP Connection (IPC) control and Network Function control services are managed by the P&P Manager as per-slice control and management tools that are deployed when needed as containerized applications following the P&P microservices approach defined in D4.1 [3]. Their deployment is subject to the slice capabilities required by the vertical, and is driven by Service & Slice Orchestrator (SS-O) at the slice instantiation phase.

Similarly, Figure 8 shows how the P&P Manager fits in the SliceNet system architecture at the DSP level. In this context, the P&P Manager covers two orthogonal features: i) lifecycle management of the P&P control instances offered to verticals, ii) lifecycle management of per-slice cognition functions. The former aspect reflects the original scope of the SliceNet P&P principles and features, as described in D4.1 and refers to the orchestration of the tailored control exposure capabilities offered to verticals through dedicated per-slice control instances. On the other hand, the latter aspect is part of the extended SliceNet P&P approach, and allows within the DSP flavour of the SliceNet management framework to activate per-slice specific cognition features, including custom machine learning algorithms, QoE monitors and QoE optimization functions, on-demand and only when needed according to the capabilities required by the vertical.

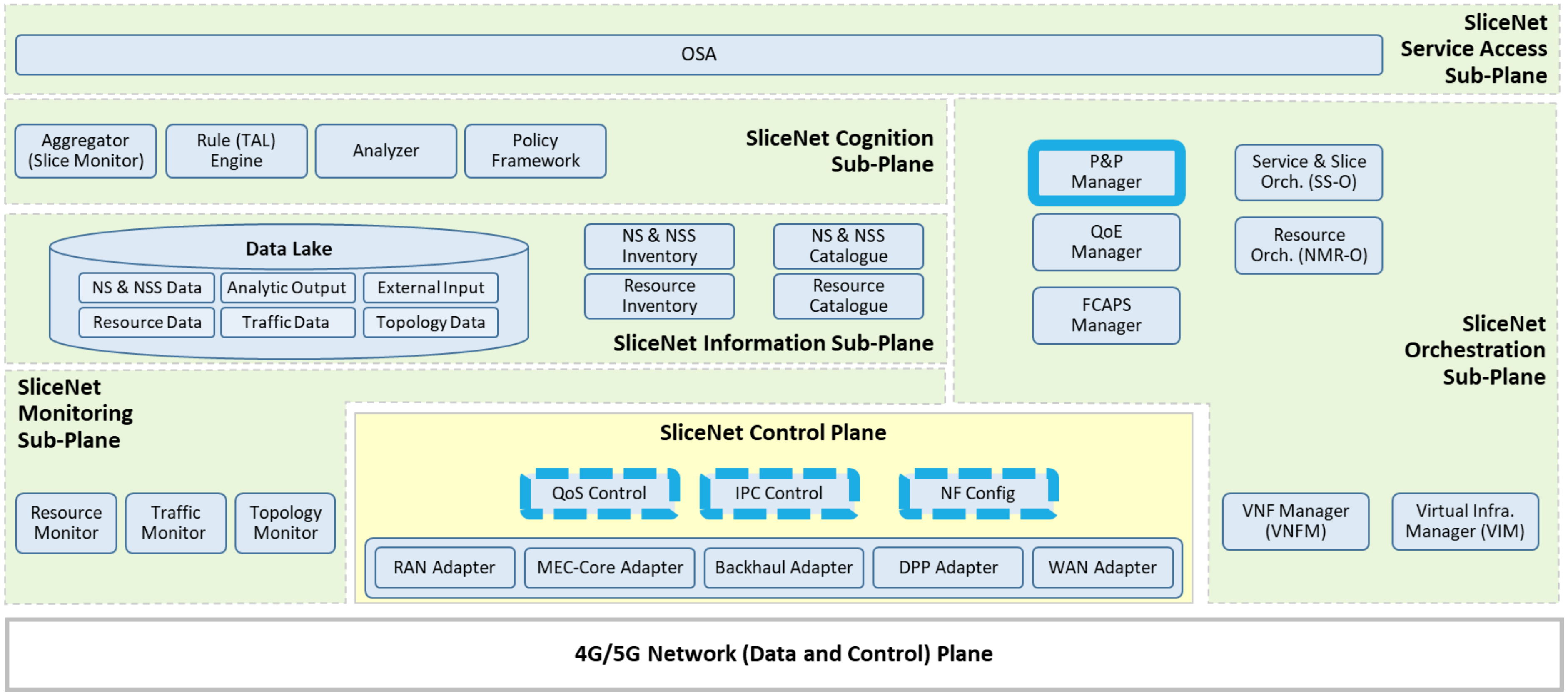


Figure 6 SliceNet system architecture – NSP level

In summary, the P&P Manager is deployed as part of the orchestration sub-plane in both NSP and DSP SliceNet cognitive management platforms, and it enables a common coordination of the whole of per-slice control and management features envisaged in SliceNet, being them either conceived to be exposed to verticals (through the P&P control plane instances) or deployed within the SliceNet platform itself. This combined (and enhanced with respect to its original definition) P&P approach allows to implement a more granular per-slice control paradigm, enabling isolation of control and management logics among different slices at the level of the SliceNet platform.

The following two subsections provide a brief summary of the main features and goals of the P&P Manager, starting from its original scope and highlighting the new introduced concepts.

3.1 Management of vertical tailored Plug & Play control

The SliceNet P&P Manager was originally conceived within the SliceNet system architecture as the lifecycle manager of the P&P control instance exposed to verticals to offer a tailored runtime control environment and customized slice view, as described in D2.4 [2]. This principle is still valid, and it refers to the following P&P features:

- activation and configuration of vertical tailored runtime P&P control functions in the form of containerized applications and creation of per-slice control environments as cloud-native Kubernetes Services [26][27]
- customization of the slice view to be exposed to the vertical through the P&P control, according to the level of control exposure required by the vertical and offered by the DSP
- customization of the vertical oriented APIs to regulate the runtime control operations on top the customized slice view, again according to the level of control exposure required by the vertical and offered by the DSP

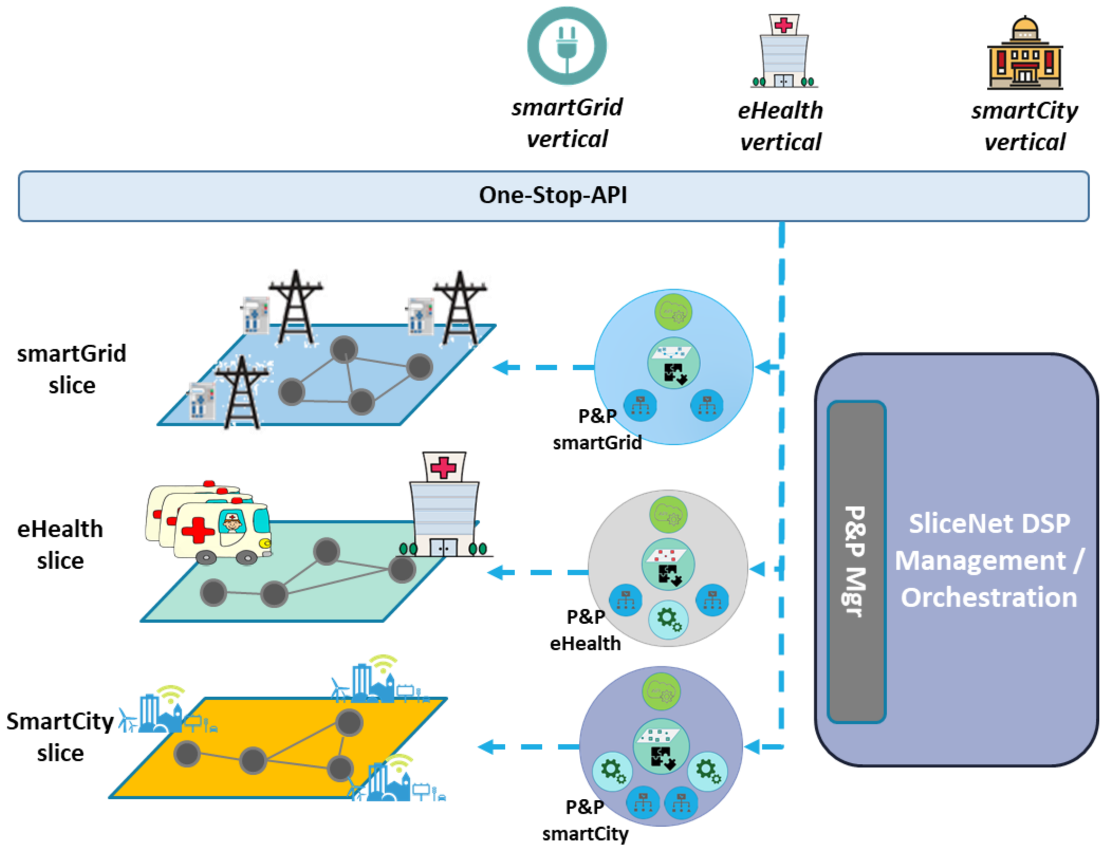


Figure 7 Management of vertical tailored Plug & Play control

This was also reflected in D4.1, where the P&P Manager is the coordinator for all the workflows and procedures related to the P&P control instances lifecycle, from activation to update and termination. In summary, these original P&P Manager features are still valid and mostly apply to its deployment in the SliceNet management platform at the DSP level. Indeed, it is at the DSP level that the per-slice P&P control instances are exposed towards the verticals through the mediation of the One-Stop-API that represents the SliceNet platform service access layer, as shows in Figure 7.

3.2 Lifecycle management domain extended: on-demand slice control and management functions

The original idea behind the P&P Manager was to provide the lifecycle management for the P&P control instances, as discussed in the previous section. In addition to this initial principle and design choice, the whole SliceNet P&P paradigm has been revised during the rest of WP4 and WP6 activities with the aim of fully benefitting of its programmable and cloud-native approach. Indeed, the adoption of a microservices-based architecture, together with the deployment of control and management applications as software containers can be easily extended and applied to other SliceNet system architecture components beyond the P&P control reported in D4.1.

In particular, the SliceNet Control Plane services already follow such kind of approach, as described in D4.3, and they have been indeed the first set of SliceNet system architecture applications that have been integrated as part of this P&P extended domain. In addition to the SliceNet control plane services, the general principle for having further control and management applications and functions to be integrated as part of the SliceNet P&P architecture is that they should match the following requirements:

- **on-demand functions**: they are required for a specific control or management purpose within a vertical service and slice context, thus they are not intended to be intended to be applicable in every scenario and they can be removed once they no longer need.
- **per-slice functions**: the functionalities they provide make sense only in the context of a given slice. In general, such functionalities are related to specific vertical requirements. For that reason, they can be created when the slice is provisioned or when the slice is already running and they are terminated at the slice decommissioning.
- **deployable as software containers:** they can be deployed as containerized applications and put under the control of a proper container orchestrator (e.g. Kubernetes [27])

A control or management application matching the characteristics listed above needs to be created, configured and updated at runtime by a proper lifecycle manager that should be able to deploy such an application in the correct context with a certain configuration. In order to avoid the design and subsequent development of a plethora of lifecycle managers, one per each family of components but with the same functionalities, the idea is to put all of those dynamic SliceNet architectural components, which presents practically the same lifecycle management requirements, under the control of the P&P Manager. One of the key features of such kind of control and management applications is that they can be containerized and managed using a proper software container orchestrator. In the solution proposed for the P&P Control and reported in D4.1, the deployment is based on a set of containers managed by Kubernetes: the P&P control instance is first deployed (as a Kubernetes Deployment) and then exposed as Kubernetes Service, in order to make it reachable from the One-Stop-API service access layer. In a similar way, additional control and management applications can be deployed using Kubernetes and, depending on them own features, they can be exposed or not as a Kubernetes service.

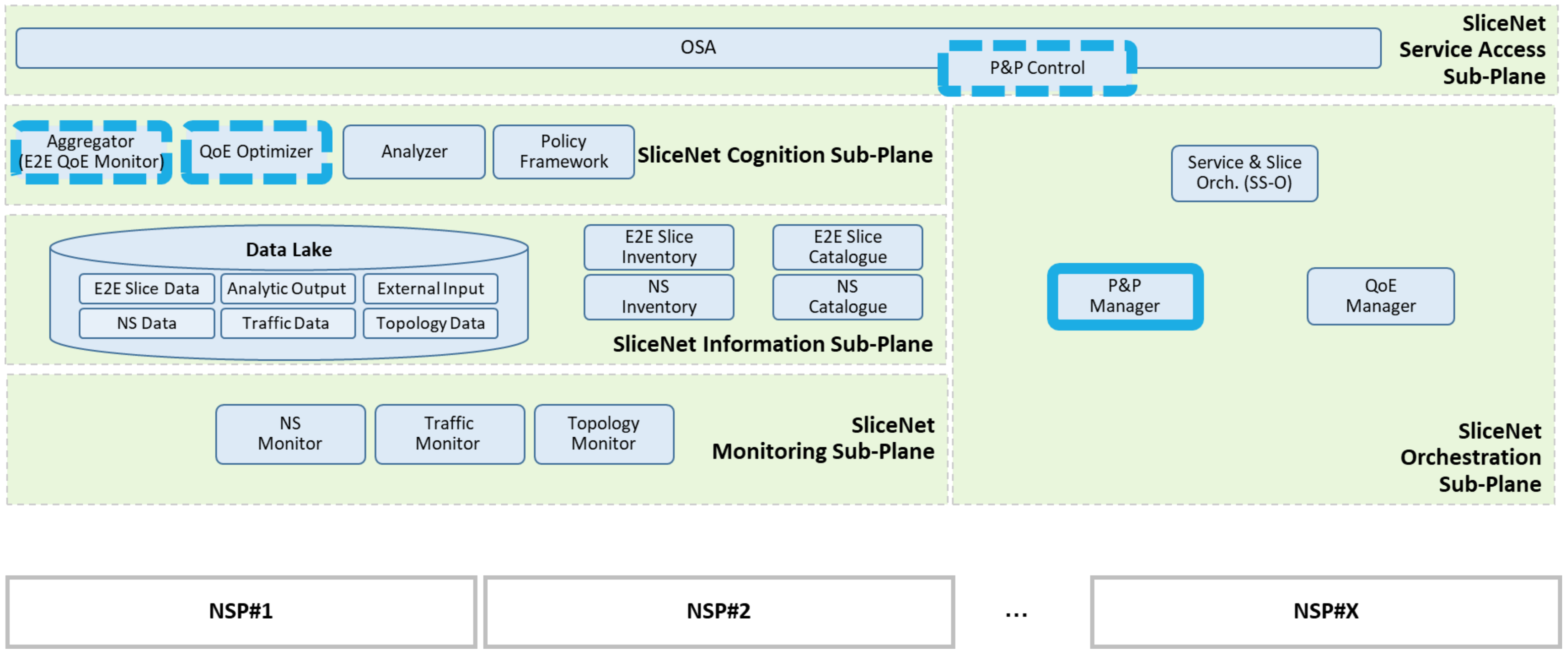


Figure 8 SliceNet system architecture – DSP level

By using the same approach already implemented and validated for the P&P Control, it possible to extend the set of the components deployed by following the vertical requirements, from a limited set of control functions, mainly oriented to offer a tailored view of the vertical slice, towards a complete set of control and management applications within the SliceNet system architecture sub-planes (Control, Monitoring, Cognition, etc.).

Therefore, from a DSP and NSP service and slice orchestration perspective, the P&P Manager provides common and unified lifecycle management of heterogeneous control and management applications, hiding the complexity of container orchestrators logics and APIs and exposing common operations for starting and stopping per-slice functions. As said, the SliceNet control plane services described in D4.3 have been already integrated as part of this new evolved P&P approach, and therefore the P&P Manager described in this document is in charge of managing the lifecycle of per-slice instances of QoS Control, IPC Control and NF Config applications within the SliceNet NSP management platform. Similarly, the QoE optimization functions part of the SliceNet cognition features at the DSP level for end-to-end service and slice optimization will be integrated as part of the evolved SliceNet P&P features, following the preliminary validation carried out in the context of deliverable D4.1.

Additional SliceNet control and monitoring applications fit the new P&P approach, and in particular the SliceNet cognition sub-plane functions at the DSP and NSP levels and the FCAPS management framework in the SliceNet NSP management platform, as described in the next two sub-sections.

3.2.1 SliceNet Cognition Sub-Plane functions: the serverless Plug & Play approach

Beyond the QoE optimization functions mentioned above, most of the SliceNet cognition sub-plane applications fit the evolved P&P approach. Indeed, data analytic functions can be natively considered as per-slice and on-demand applications in the context of a management framework. Moreover, they can normally be easily packetized as software containers.

More than that, per-slice data analysis is typically triggered by specific events or conditions, resulting therefore into functions and applications that can be considered as short or medium lived and not run for the whole lifetime of a given slice. In this context, the Functions-as-a-Service (FaaS) paradigm fit very well and can be leveraged to further enhance the SliceNet P&P paradigm towards an innovative cloud-native architecture where per-slice control and management containerized applications are combined with serverless functions that are executed only in response to slice specific conditions or events. FaaS is the commonly used and more descriptive name for the core of serverless computing offerings from the public cloud providers. FaaS allows to execute applications (or even code snippets) on-demand based on user-defined events, and it is most suited for short-lived applications. Basically, a FaaS platform embed orchestration features that monitor the triggering events, allocate a runtime (that normally is a container) for the function, executes it, and persists the results. Commercial solutions such as Amazon Lambda [28] offer FaaS platforms in the cloud for a wide plethora of applications, from data analytics to IoT and web services.

For the purpose of its QoE end-to-end monitoring and data analysis based on Machine Learning techniques, SliceNet integrates FaaS within its P&P architecture to make even more programmable and flexible the per-slice control and management logics, enabling serverless applications to be deployed as part of the SliceNet cognitive platform at both NSP and DSP levels. In particular, the SliceNet P&P architecture makes use of OpenWhisk [29] as serverless platform and leverages on its native integration with Kubernetes to combine regular long-lived per-slice control applications (like the SliceNet control plane ones) with FaaS-enabled short-lived cognitive applications (like those for fault prediction under development in WP5).

In summary, the P&P Manager coordinate the deployment within the SliceNet management platform (at both DSP and NSP levels) of an heterogenous set of per-slice and on-demand control and management applications, hiding the complexity of their deployment and configuration through a common set of APIs exposed towards SliceNet service and slice orchestration functions, irrespectively of the nature of the P&P application (i.e. regular containerized applications vs. short-lived elastic serverless functions), as described in sections 4 and 5.

3.2.2 Relation with the SliceNet FCAPS Management Framework

The SliceNet Fault Configuration Accounting Performance Security (FCAPS) Management Framework is providing the means for enabling automation regarding the management loops that can ensure the maintenance of the agreed service level agreement and slice capabilities and requirements by detection of situations that have to be given particular attention and enforcement of foreseen reactions or remedies [30]. In this context the processing that is applied is instantiated per slice according to the resolution of the service requirements as these have been negotiated between DSP and verticals. The resolution is subject to the monitoring and actuation options supported by the NSP. Once the monitoring and actuation options are clarified per network function deployed within the given slice that is to be utilized in the domain of one NSP, the FCAPS Framework activates and customizes the appropriate artefacts that will produce the required monitoring information. On top of this information, rules or policies are applied to allow for automated reaction when the abovementioned situations are detected. The reactions include invocation of actuation options among those being offered by the NSP. In this respect the FCAPS Framework follows the SliceNet orchestration process by detecting the slice details as those are related with appropriate sensing and actuation descriptors. Such functionality is currently integrated as one instance per NSP dealing with the overall SliceNet catalogue and inventory information that is maintained. Additionally, the collected information is stored in an NSP specific database for all slices and the Rule Engine that processes the automation policies maintains a pool of processing objects for all slices.

Aiming at producing a concrete framework that can enable the separation of roles as implied by the relevant 3GPP model, the single FCAPS Framework instance per NSP approach is under evaluation regarding the level of isolation and granularity that is achieved in the context of multidomain slicing. It is expected that during the final period of the project the concepts will be challenged not only from the technical point of view but also from the administrative point of view. Therefore, in case the single FCAPS Framework instance per NSP is considered as not adequate enough to be claimed as enabler for slice isolation, the approach can be re-engineered based on the capabilities of the SliceNet P&P framework that can support the provisioning of one instance of the FCAPS framework per slice inside the same NSP. In such a case the FCAPS Framework setup will be provided following the P&P practices so that it can be activated on the fly every time the underlying set of Network Functions is orchestrated.

4 Plug & Play Manager in SliceNet

The main role and responsibilities of P&P Manager has been already introduced in the previous sections and described in D2.4. This chapter provides a description of the modules building the P&P Manager high-level architecture as well as the interfaces exposed and the functionalities offered to the Service and Slice Orchestrator (SS-O).

4.1 High level architecture

Figure 9 depicts a representation of the architecture of the P&P Manager (as presented in D2.4), highlighting the main modules responsible for the management of the lifecycle of the various per-slice application instances. Although the high-level design and functional decomposition of the P&P Manager follows the same definition as in D2.4, the implementation and features of the modules has been subject to several design modifications and updates as described in Section 3.

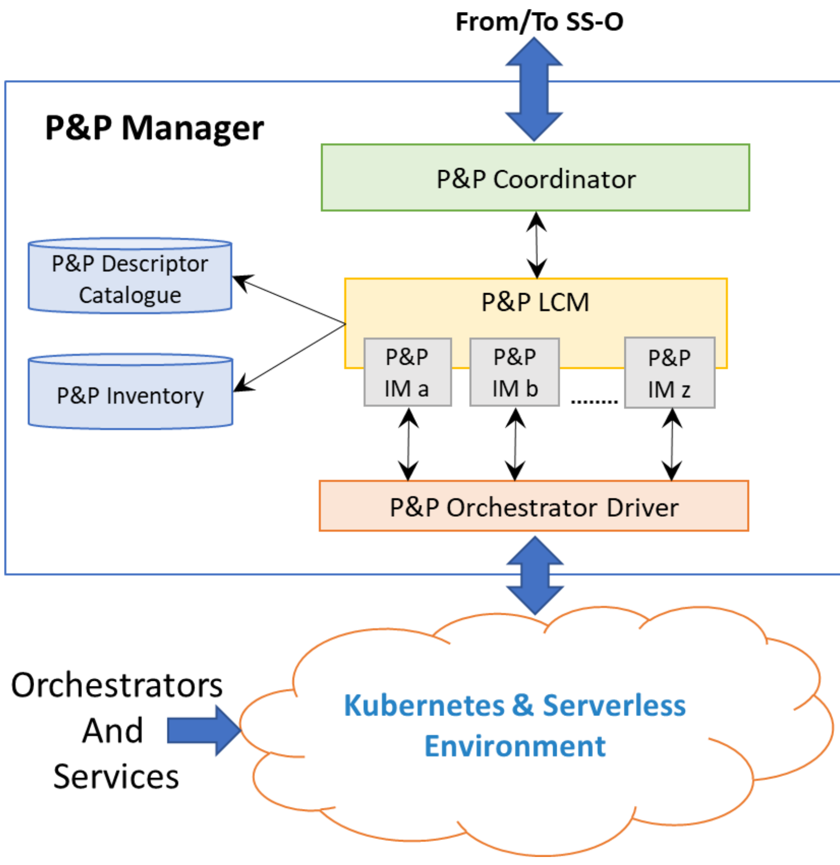


Figure 9 High-level P&P Manager architecture

In this sense, the role of the modules, with respect to one described on D2.4, has been extended for addressing the new requirements.

The ***P&P Coordinator*** represents the entry point of the P&P Manger. It offers all of the function needed for the base lifecycle management of the per-slice application instances (create, update, terminate, retrieve information), as well as a set of administrative functions, exposed for the management of P&P manager itself (e.g. update Inventory and/or Catalogue). The SS-O at both NSP and DSP levels is the consumer for the interface exposed.

P&P Lifecycle Manager (P&P LCM) is the core module for P&P Manager. It is responsible for parsing the requests sent by the SS-O, through the P&P Coordinator, decomposing and transforming them in a sequence of actions for creating, updating and terminating each per-slice application instances. In order to do this, the P&P LCM creates proper instances of *P&P Instance Manager* (described below) that actually handle all of the modules deployed for a given slice. The P&P manager exploits the interface exposed by each P&P IM in order to be able to access both to the Container Orchestrators (for actual life cycle operations) and the per-slice application instances themselves for configuration at runtime. In this sense, the P&P LCM implements the actual logic for the management of the instances while the P&P IM keeps them status.

P&P Instance Manager (P&P IM) manages all the components created per slice instance. It keeps track of the status of the instances in terms of composition of the set of modules and in terms of single module status. It consumes the interfaces exposed by the driver for the interactions with the orchestrators and the modules instantiated.

P&P Driver (formerly P&P Configuration Driver) is the module that actually communicate with the per-slice application instances and the Container Orchestrators. It offers a common access interface for the P&P IMs and thus hide the specific complexities of heterogeneous environments (i.e. Kubernetes and OpenWhisk).

P&P Descriptor Catalogue (formerly P&P Drivers Catalogue) As already described, the modules, deployed at runtime, follow an approach based on the software containerization. Originally, the store should have contained the base images for such containers but, in order to provide more flexibility, the store actually contains special descriptor files that indicate, among the others, also where the container image can be found. In this way, different kind of object (e.g. Docker images or FaaS functions) can be described in the same catalogue.

P&P Inventory contains all of the information needed for the deployment and the configuration of the per-slice applications. Such information is generated by proper algorithms implemented in P&P LCM taking into account the requests from the SS-O and the information stored in the P&P Descriptor Catalogue.

Beside the roles and responsibilities of each functional component, the P&P Manager has been designed as a modular software in which a specific module is devoted to a specific task. The modularity of the P&P architecture can be considered a key concept that made the software easy extendable, allowing the control of the lifecycle of new families of per-slice applications becoming part of the SliceNet on-demand control and management framework a straightforward development job. Originally, as widely discussed, the P&P Manager has been designed for managing the P&P Control instances only and, when other types of application have been put under its control, the P&P Manager itself has been enhanced in order to fulfil the new requirements for the management of the new lifecycles: this is what happened for all of those P&P applications already mentioned (CPSs, QoE optimizer, etc.) that are were not part of the original P&P Control scope defined in D4.1.

The common characteristics among all per-slice application families, as mentioned in Section 3, have eased their integration process and reduced the effort in extending the P&P Manager software. Indeed, most of the software modules developed for the lifecycle management of P&P Control instances (in its original scope) have been reused either “as is” or after slight upgrades (e.g. new data structures for tracking the instances, a new table on the DB, etc.).

In practice, the main enabler of this P&P Manager extendibility is the P&P Driver internal function. One of the main issues in integrating new families of per-slice applications is indeed to provide a mechanism for interacting with the proper Container Orchestrators and communicating with them once they have been instantiated, a task in charge of the P&P Driver. As described in Section 5, the P&P Driver is not a monolithic module but, on the contrary, it is implemented as a multi-module element with pluggable clients. Currently, two client modules are available: a REST client (for configuring the P&P Control instances) and a Kubernetes client. Such clients are fully reused and leveraged for the lifecycle management of the additional P&P per-slice applications covered with this document: both CPSs and QoE optimizer have been containerized and are deployed by exploiting Kubernetes as container orchestrator [27] (exactly as the P&P Control applications), while OpenWhisk [29], the FaaS orchestration facility, exposes a REST interface that can be consumed by the P&P Driver REST client.

Given the pluggable approach implemented within the P&P Driver, if another family of per-slice applications has to be managed with the P&P Manager, a new client module is required without any effect on the core components of the P&P Manager, that will still make use of the common interface exposed by P&P Driver.

4.2 Workflows and mechanisms

The lifecycle of each per-slice application is strictly related to the lifecycle of the network slice itself. Once a network slice is created, all of the per-slice applications needed should be deployed and configured (at creation or during the slice lifetime) and once the network slice is terminated, all of the related per-slice applications allocated should be in turn destroyed.

In the light of above, the triple *create-configure-terminate* represents the minimum set of functions required for managing the lifecycle manager of the control and management applications deployed per-slice, as it represents also the minimum set of functions for managing the lifecycle of the slice itself. For sake of completeness, it is worth noting that the *update* function is also available and applicable to all of the applications managed by the P&P Manager. In general, the update operation can be applied also to the slice itself: a modification at runtime of the per-slice application set is a concrete example of slice update.

4.2.1 Per-slice application creation

The creation process for one or more per-slice application is always triggered by the SS-O invoking the proper API on the interface of the P&P Manager. P&P Manager elaborates the request from the SS-O, checks for proper application to be instantiated in the P&P Catalogue and generates ad hoc deployment and configuration files. A P&P IM instance is created for the given slice (thus managing the lifecycle of the whole set of per-slice control and management applications) and instructed to invoke the creation command on the interface exposed by the proper Container Orchestrator(s).

The create operation can be triggered in two different phases of the slice lifetime: i) at slice creation and ii) at runtime, when the slice is already up and running. In the former case, when a new slice is deployed, all of the control and management applications belonging to it must be created. In the latter case, the SS-O can invoke the creation process for adding further components not included at slice creation time: this case falls into the slice update operation.

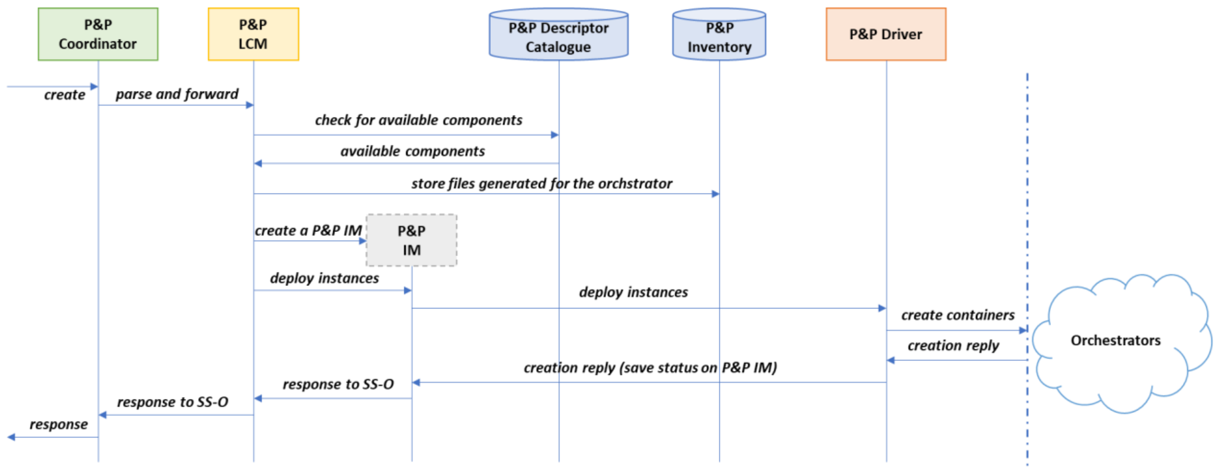


Figure 10 Create operation sequence diagram and P&P Manager involved modules

The sequence diagram shown in Figure 10 describes the internal operations and the module involved of P&P Manager during the create process. The diagram is referring to a successful create operation. In the case of the creations of instances has no success, all the already deployed applications are cleaned automatically by a rollback operation and an error message is sent to the SS-O.

4.2.2 Per-slice application configuration

Once the per-slice applications have been deployed, they need to be configured, in order to properly interact with the other components in the SliceNet cognitive management platform. For that reason, the P&P Manager provide, through its northbound interface, a mechanism and an API for configuring the per-slice application instances that are actually running. Figure 10 describes the sequence of actions happening inside the P&P Manager when the SS-O invokes the configure operation.

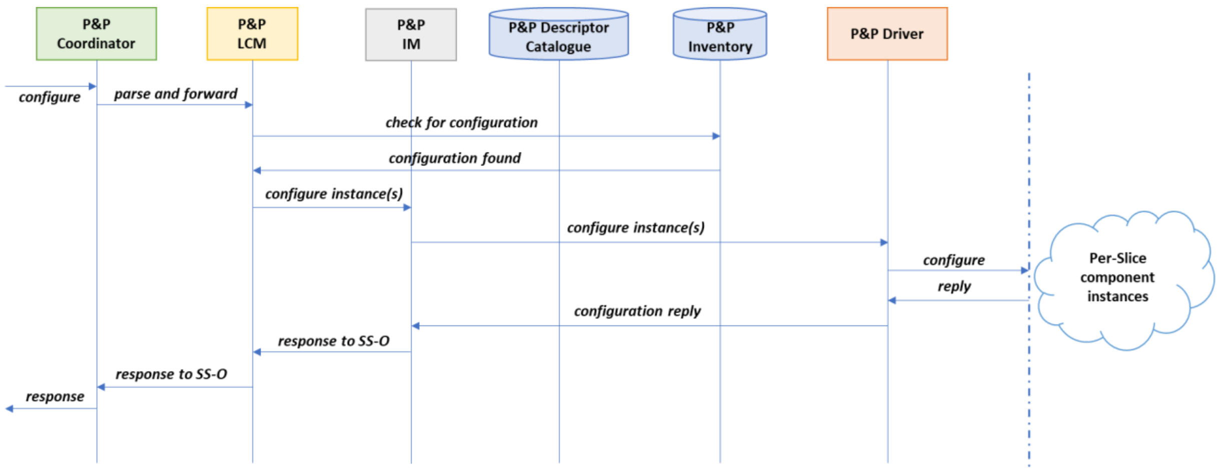


Figure 11 - Configure operation sequence diagram and P&P Manager involved modules

It is worth nothing that the API exposed by P&P Manager does not represent the only way for configuring the running instances. Indeed, this operation can be performed also in an automatic way both from the P&P Manager and from the application itself, especially at application's boot time. For example, a given per-slice application instance could not need any kind of runtime configuration: in this case, a proper configuration file put inside the software container could provide all the information it needs for working properly.

Another way is to generate the files representing the input for the Container Orchestrators (for deploying the per-slice applications) by including also the configuration needed by the applications themselves: such information will be available to the applications at boot time.

4.2.3 Per-slice application termination

Figure 12 describes the P&P Manager internal steps when the SS-O invokes the termination of the slice. This operation can be applied at the whole slice level, thus bringing to the termination of all the relevant per-slice applications deployed within the SliceNet cognitive management framework, as well as at a per-application level, and in this case, it can be considered as a runtime update of the given slice.

Once the P&P Manager receives the terminate request for a given slice, it forwards such request to the proper Container Orchestrator(s) through the P&P IM that actually handle the set of application instances. After the instances have been terminated, the P&P LCM terminates the P&P IM that no longer needs and remove any generated file related to the removed slice form the P&P inventory. Finally, a notification is sent towards the SS-O through the P&P Coordinator to acknowledge the operation.

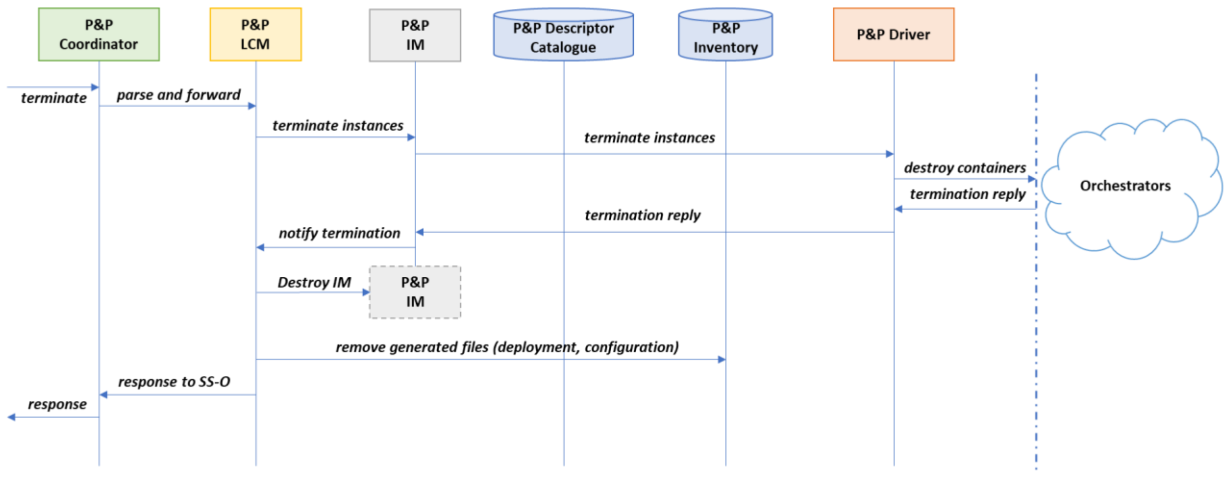


Figure 12 - Terminate operation sequence diagram and P&P Manager involved modules

4.2.4 Per-slice application update

During the slice lifetime, according to evolving requirements in terms of network slice characteristics impacting its control and management logics at either DSP or NSP level, the SS-O is responsible of dynamically update an existing slice by creating or removing (or even re-configuring) per-slice applications within the SliceNet cognitive management platform, as described in the two previous sections. This operation can be also used to update an existing control and management application to a newer version, if applicable.

4.3 Information model and interfaces

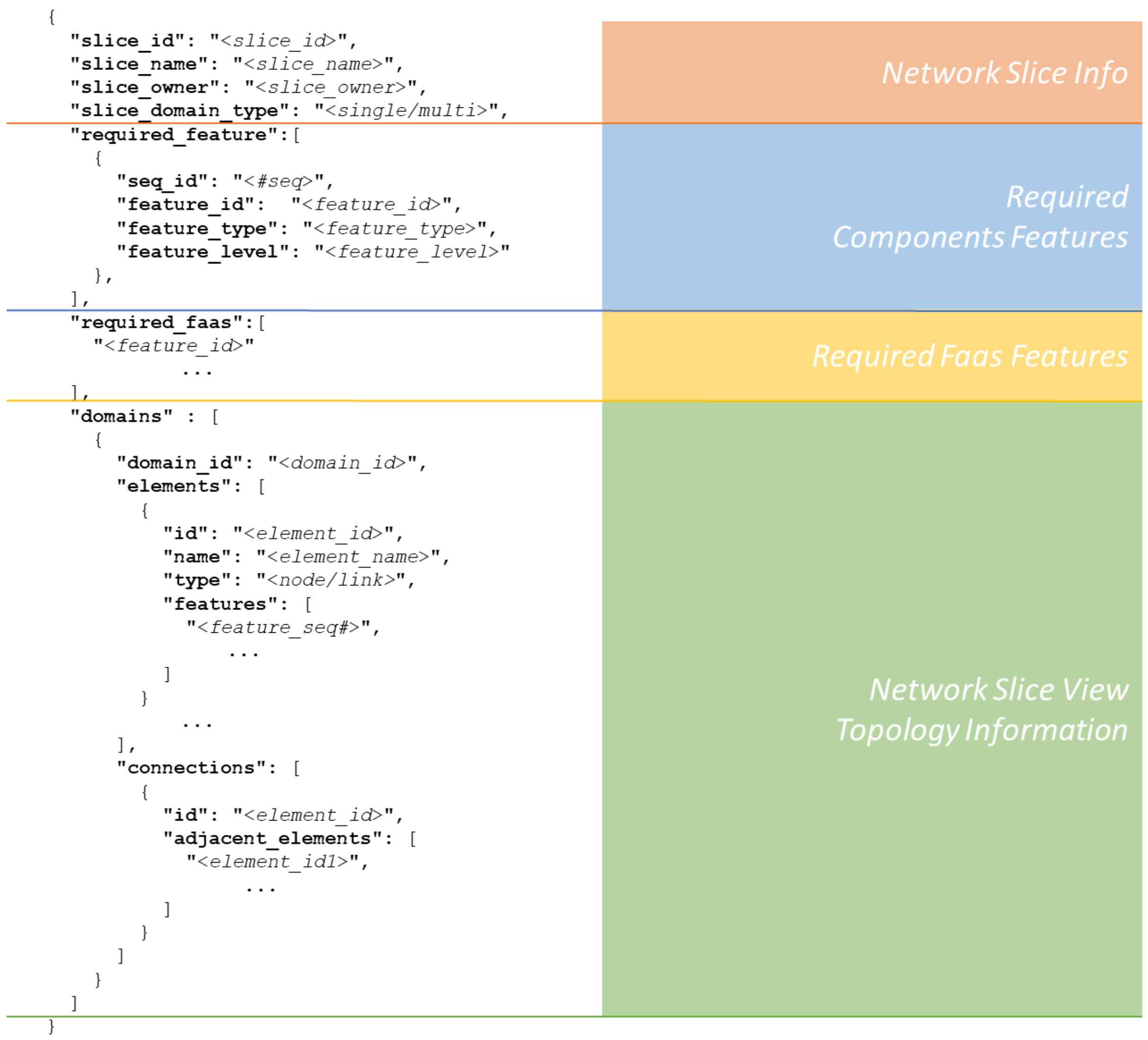
The P&P Manager exposes its own northbound interface (NBI) towards the SS-O, the only component within the SliceNet system architecture that is allowed request the deployment of per-slice control and management applications. At the DSP level, the set of per-slice applications to be deployed depends on the requirements specified from the vertical at network slice creation and update. Such requirements are elaborated from the SS-O and condensed in a deployment request the P&P Manager is able to consume. In a similar way, at NSP level, the SS-O maps the network slice requirements expressed by a given DSP in order to produce a slice creation request for the P&P Manager and deploy the proper set of per-slice control and applications.

4.3.1 Plug & Play Manager information model

In this section a formal description of the information model that the P&P Manager uses for managing the lifecycle of the set of per-slice applications is defined. Such model, detailed in the following tables, includes the information needed for deploying the set of per-slice applications needed to control and manage a given network slice and to describe the requirements of each control and management application.

4.3.1.1 Plug & Play deployment information model

Figure 13 shows the information model for the set of data the SS-O sends to the P&P Manager for creating a new set of on-demand and per-slice control and management applications. Such model consists of four main parts providing respectively general network slice information, required components features, required FaaS features (if any) and network slice view topology information.



```
{
  "slice_id": "<slice_id>",
  "slice_name": "<slice_name>",
  "slice_owner": "<slice_owner>",
  "slice_domain_type": "<single/multi>",
  "required_feature":[
    {
      "seq_id": "<#seq>",
      "feature_id":  "<feature_id>",
      "feature_type": "<feature_type>",
      "feature_level": "<feature_level>"
    },
  ],
  "required_faas":[
    "<feature_id>"
           ...
  ],
  "domains" : [
    {
      "domain_id": "<domain_id>",
      "elements": [
        {
          "id": "<element_id>",
          "name": "<element_name>",
          "type": "<node/link>",
          "features": [
            "<feature_seq#>",
                ...
          ]
        }
            ...
      ],
      "connections": [
        {
          "id": "<element_id>",
          "adjacent_elements": [
            "<element_id1>",
                 ...
          ]
        }
      ]
    }
  ]
}
```

Network Slice Info
Required Components Features
Required Faas Features
Network Slice View Topology Information

Figure 13 – Template of P&P deployment Information data sent by the SS-O

The ***Slice Info*** section contains general information for identifying the network slice (e.g. the slice_id) in support of which the control and management applications have to be deployed within the SliceNet cognitive management platform. It consists of four fields described in Table 1.

Identifier	Cardinality	Description
Slice_id	1	Id of the slice expressed as Universally Unique Identifier (UUID)
Slice_name	1	Name of the Slice chosen from the Vertical or the DSP (e.g. `smart-grid-001`)
Slice_owner	1	Owner of the Slice (e.g. `Efacec`)
Slice_domain_type	1	Indicates if the network slice is a single or multi-domain one. Possible values are SINGLE or MULTI.

Table 1 Slice Information fields in SS-O CREATE request

The ***Required Slice Features*** represents the second section of the P&P deployment information model and consists of a list of elements that represent the per-slice control and management features requested by the SS-O and that the P&P Manager should deploy as applications. Such features are intended to cover heterogeneous domains and can belong to different applications for control, management and monitoring. Table 2 and Table 3 details how these required features are structured.

Identifier	Cardinality	Description
required_feature	1	List of the control and management features required for the given network slice. See Table 3 for individual elements in the list

Table 2 List of KPIs elements in SS-O slice information model

An example of feature, offered by a given application, could refer to monitoring information about the throughput of the User Equipments (UEs) belonging to the network slice. Such feature needs to be formalized in order to make the P&P Manager able to understand which applications are required to support and provide the functionality.

Table 3 shows the elements characterizing a *required_feature* element. The key idea is that the same *feature_id* can be associated to different *feature_level* as well as to different *feature_type*. As an example, a feature with *feature_id* "QoS_handle" can be applied with different granularity (e.g. end-to-end level as well as overall slice) and with different scopes (e.g. monitoring or control). For that reason, a feature is uniquely identified by the triple *feature_id - feature_level - feature_type.*

Identifier	Cardinality	Description
seq_id	1	Feature sequence number in the list. This number is used for associating the feature to one or more elements of the network slice view topology information set
kpi_id	1	Name of the feature.
feature_level	1	Represents the abstraction level of application of the feature. Currently 3 levels are available: • **element:** the feature is associated to single elements of the network slice view topology (Vertexes or edges in the topology graph) • **e2e:** the feature is associated to an end-to-end communication (e.g. UE-EPC) • **slice:** the feature is associated to the slice as a whole (e.g. monitoring of the slice throughput)
feature_type	1	Provides the scope of the feature: Management, Monitoring or Control

Table 3 Element of KPI List in the Slice Information Model

In the light of above, coming back to the UE monitoring example above, we could consider the following triple:

- *feature_id*: ***QoS_monitoring***
- *feature_level:* ***Element*** *(it is a node of the network slice view topology graph)*
- *feature_type:* ***Monitoring***

This set of information related to the required control and management features is then processed by the P&P Manager to select and deploy the proper per-slice applications. In particular, in the case that a particular feature is supported and implemented by different applications (e.g. regular containerized and serverless) the P&P Manager can decide, according to predefined policies, which one to select.

The third section in the P&P deployment information model is a list containing the ***Required FaaS Features,*** and it has to be considered optional and to be used only when the SS-O itself wants to force the deployment of specific serverless applications**.** As shown in Table 4, the elements of the list are the IDs of the functions the SS-O requires to be loaded at the FaaS facility (i.e. OpenWhisk).

Identifier	Cardinality	Description
required_faas	0..1	List of the *feature_id* to be loaded on the FaaS facility

Table 4 List of feature_ids required in SS-O slice information model

The last section of the P&P deployment information model is the network slice view ***Topology Information*** and, as the name says, provide topological information about the network slice and it is used for building the vertical oriented network slice customized view at the DSP level only. The concept of slice topology, intended as vertical view of the slice is widely covered in D4.1, where different models have been presented. Table 5 describes the entry for the Topology Information sections that is a list of topologies grouped by domains.

Identifier	Cardinality	Description
domains	1	List of topological information needed for building the network slice view topology graph at the DSP level (See D4.1). If the list contains more than one element, it implicitly means that the slice is a multidomain one.

Table 5 List of topological information object in SS-O slice information model

Indeed, Table 6 describes the topological information object structure, in which can be found the *domain_id*. Such objects represent the items of the list in Table 5.

Identifier	Cardinality	Description
domain_id	1	Id of the Administrative domain
elements	1	List of node/link composing the slice topology view graph

Table 6 Topological information Object belonging to the "domains" list

The other field in Table 6 is *elements* that is a list of objects described in Table 7.

Identifier	Cardinality	Description
id	1	Id of the element
name	1	Name of the element
type	1	Type of element: NODE or LINK
features	1	List of required control and management features per element, identified by seq_id defined on Table 3

Table 7 Structure of an element belongs to the Topological information object

As described above, each Element has associated a list of features representing the slice control and management features the SS-O requires for that node or link in the topology view, and indicated as a list of sequence numbers. As in the SS-O request a feature is listed only once, the sequence number identify the correspondent KPI univocally and is hence enough for associating features to an element without ambiguity. A special element with the *id* set to the *slice id* is used for specifying overall slice features. This is used as default option in the case of P&P Manager deployed at NSP level where all the per-slice control and management applications have a slice-wide scope.

The *connections* field in Table 6 is an adjacency list that specify the interconnections between the topological elements in the customized view (nodes and links). It is applicable at the DSP level only.

Identifier	Cardinality	Description
id	1	Id of an element
adjacent_elements	1	List of adjacent elements

Table 8 – Structure of the item belonging to the Adjacency list

As discussed above, the slice topology, intended as slice view tailored for the vertical, has been introduced by the P&P Control in D4.1, in order to offer towards the vertical an abstraction of the slice elements represented as an interactive graph: the vertical can play with elements that exposes some control and management feature (e.g. get the throughput of an UE) through the One-Stop-API access layer. The reason why the concept of slice topology view can be found also in the P&P Manager (and in SS-O) is due to the fact that for building such detailed view graph, the P&P Control needs information that it cannot know a priori. In few words, the building of the slice view is a 3-step process, starting from the SS-O and terminates in the P&P Control. From the topology information (including the required features) received by SS-O (step 1), the P&P Manager finds the correspondent components descriptors (See 4.3.1.2) in its own inventory and maps the features to the correspondent functions (i.e. available control and management applications). From there, the P&P Manager generates a slice view to send to the correspondent P&P Control instance (step 2), following the procedures and mechanisms defined in D4.1. The P&P Control instances augments the received slice view data by associating proper APIs from the interfaces of the deployed plugins, in order to create the final slice view consumed by the One-Stop-API service access layer (step 3).

It is important to highlight that a deployment request from the SS-O might not contain all of the information presented in this section. In general, taking into account the four parts of the data model presented, the only two that are mandatory in the request are the Slice Info and the Required Features for, at least, a couple of reasons:

1. *Position of the P&P Manager in the SliceNet architecture*: if the P&P Manager is positioned at the NSP level, there is no need of the Topology information since there is no need of a customized slice view towards the vertical;
2. *FaaS not needed*: not all of the network slice instances require control and management applications to be implemented as serverless functions

In conclusion, while the P&P Managers deployed at DSP and NSP level follows the same functional architecture (as shown in Figure 9) and are implemented by the same software prototype (as described in section 5), the *create slice requests* they receive and handle does not contain the same type of requirements and information as in the NSP case there is no direct interaction with verticals (and direct exposure of tailored slice view) .

4.3.1.2 Component Descriptor information model

The following sequence of tables will describe what are the elements building a Component Descriptor, that capture the capabilities of available per-slice control and management applications. The scope of the descriptor is to provide information about the component, in terms of deployment (e.g. container image) and capabilities, which are the set of features the component is able to offer for a give network slice.

Identifier	Cardinality	Description
Component_id	1	Unique identifier of the component
Component_name	1	Name of the component
component_ports	1	List of TCP ports the service provided is available
Component_image	1	Reference / identifier of container image of the component

capabilities	1	List of features offered by the component in the nested format:
		`"<feature_type_value>": {`
		`    "<feature_level_value>": [`
		`      {`
		`        "<feature_id>": "<comp_feature_id>"`
		`      }`
		`    ]`
		`  }`
		feature_type_value, feature_level_value, feature_id are the same values used for identifying the feature in Table 3 .
		comp_feature_id is the correspondent feature id offered by the component

Table 9 – Elements of the Component Descriptor

In Table 9 are shown the elements of a component descriptor. The field *capabilities* maps the features defined as SS-O does with the correspondent features offered by the components. Basing on such information, the P&P Manager is able to find a set of elements (P&P functions, Control plane services, serverless functions, etc.) to be deployed for satisfying the request of the SS-O. Since the descriptor are formatted as JSON files, in order to speed up the lookup process, the P&P Manager rely on a lightweight database for improving the file indexing, as discussed in the prototype at Section 5.

4.3.2 Plug & Play Manager APIs

The P&P Manager exposes its own interface through a set of REST APIs offering different kind of features towards both the SS-O and the System Administrator. In particular, it is possible to distinguish two different set of REST APIs:

- Slice Component Set lifecycle APIs, a set of endpoints exposed towards the SS-O for controlling the lifecycle of per-slice applications.
- Management APIs, developed for SliceNet system administrators. They provide a set of operation for administrative purposes on the P&P manager (add/remove descriptors, database-rebuilding, etc)

Despite the amount and variety of the components under the management of the P&P Manager, both API sets offer a limited number of operations for performing the lifecycle actions.

Indeed, in order to keep the interface as simple as possible, the lifecycle interface exposes a set of APIs allowing the SS-O to perform only the operation described in Section 4.2.1: *CREATE, CONFIGURE, TERMINATE* and *UPDATE*.

The detail of such operations will be described in the tables below.

Endpoint	`/plug-and-play-manager/slice/<slice_id>/`
HTTP Verb	POST
Description	Deployment of a new per-slice control and management application set
Caller	SS-O
Request Body	JSON object containing the full information about the deployment requirements the P&P Manager needs for instantiating components. The structure of the request is described in Section 4.3. An example is available on Annex A.
Response Body	None
Response Codes	200 OK - CREATE operation has been executed successfully

Table 10 P&P Manager CREATE API details

Endpoint	`/plug-and-play-manager/slice-config/<slice_id>/`
HTTP Verb	POST
Description	Send the configuration to a given per-slice application instance. This method has been kept as part of the original design of the REST interface, when the P&P Manager was supposed to manage only the P&P control instances exposed to verticals at the DSP level.

Caller	P&P Manager
Request Body	A JSON object containing the P&P Control configuration as defined in D4.1
Response Body	None
Response Codes	200 OK - The P&P configuration has been successfully stored in the P&P core

Table 11 P&P Manager CONFIGURE API details

Endpoint	`/plug-and-play-manager/slice/<slice_id>/`
HTTP Verb	DELETE
Description	Termination of all the per-slice control and management applications supporting the slice <slice_id>
Caller	SS-O
Request Body	None
Response Body	None
Response Codes	200 OK - Slice applications have been terminated with success

Table 12 P&P Manager CONFIGURE API details

Endpoint	`/plug-and-play-manager/slice/<slice_id>/`
HTTP Verb	PUT
Description	Update an existing set of per-slice applications for a given <slice_id>
Caller	P&P Manager
Request Body	An update of the JSON object used during the CREATE operation for the slice
Response Body	None
Response Codes	200 OK - UPDATE operation has been executed successfully

Table 13 P&P Manager CONFIGURE API details

As discussed, in addition to the APIs offered to the SS-O, another set of operations is available for System Administrators in form of REST APIs.

Endpoint	/plug-and-play-manager/admin/command/init-db
HTTP Verb	POST
Description	Rebuild the database entries basing on current component descriptors stored on the P&P Descriptor Catalogue. P&P Manager database is described in Section 5
Caller	System Administrator
Request Body	None
Response Body	None
Response Codes	200 OK - DB rebuilt successfully

Table 14 - P&P Manager init-db operation details

Endpoint	/plug-and-play-manager/admin/command/get-db
HTTP Verb	GET
Description	Retrieve the information stored in the database
Caller	System Administrator - SS-O
Request Body	None
Response Body	A JSON object structured as follows: `{ "<table_1>": }`
Response Codes	200 OK

Table 15 - P&P Manager get-db operation details

Endpoint	/plug-and-play-manager/admin/command/set-feature
HTTP Verb	POST
Description	Set a new feature in the database
Caller	System Administrator

Request Body	`"feature_info": {`
	`      "feature_id": <feature_id_value>,`
	`      "feature_type": <feature_type_value>,`
	`      "feature_level": <feature_level_value>,`
	`    }`

Response Body	None
Response Codes	200 OK - New feature set

Table 16 - P&P Manager set-feature operation details

Endpoint	/plug-and-play-manager/admin/command/set-descriptor
HTTP Verb	POST
Description	Set a new descriptor on DB and save file on P&P Descriptor catalogue
Caller	System Administrator
Request Body	A JSON object representing the component descriptor as defined in Section 4.3
Response Body	None
Response Codes	200 OK - New Descriptor set

Table 17 - P&P Manager set-descriptor operation details

Endpoint	/plug-and-play-manager/admin/command/set-config
HTTP Verb	POST
Description	Set base configuration values for the P&P Manager (e.g. ip/port of Kubernetes master node)
Caller	System Administrator

Request Body	A JSON object consisting of the P&P Manager configuration info.
	An example:
	`{`
	`  "parser": {`
	`    "descriptor_path":`
	`"manager/manager_persistence/descriptor_catalogue/",`
	`    "core_file_name": "core_control_descriptor.json"`
	`  },`
	`  "db_handler": {`
	`    "db_name": "catalogue_db",`
	`    "db_path": "manager/manager_persistence/"`
	`  },`

```
  "slice_config_generator": {
    "base_deployment_model": "../models/deployment_template.yaml",
    "base_service_model": "../models/service_template.yaml",
    "yaml_save_path":
"manager/manager_persistence/configuration_repo/kubernetes/",
    "json_save_path":
"manager/manager_persistence/configuration_repo/slice_view/"
  },
  "cpsr": {
    "cpsrIP": "172.17.0.2",
    "cpsrPort": "8080"
  },
  "deployment_node": {
    "ip": "10.0.8.28"
  },
  "core_endpoints": {
    "cpsr": "/plug-and-play-test/pp_management/registration/cpsr-info/",
    "slice_view": "/plug-and-play-test/pp_management/registration/slice/"
  },
  "faas_api": {
    "faasIP": "10.0.8.28",
    "faasPort": "443",
    "user": "23bc46b1-71f6-4ed5-8c54-816aa4f8c502",
    "password":
"123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP",
    "namespace": "guest",
    "base_url": "/api/v1/namespaces/",
    "base_package_model": "../models/faas_package_template.json",
    "base_action_model": "../models/faas_action_template.json",
    "code_file_path":
"manager/manager_persistence/configuration_repo/faas/"
  }
}
```

Response Body	None
Response Codes	200 OK - New P&P Manager configuration set

Table 18 P&P Manager set-config operation details

5 Plug & Play Manager software prototype

The P&P Manager has been designed and developed in the context of the activities of T6.3 with the original scope of managing the lifecycle of the P&P control instances (described in D4.1). During the task activities, the management domain has been extended toward other types of components and applications, again based on Docker Containers and orchestrated by proper Container Orchestrators, as described in Section 3.2.

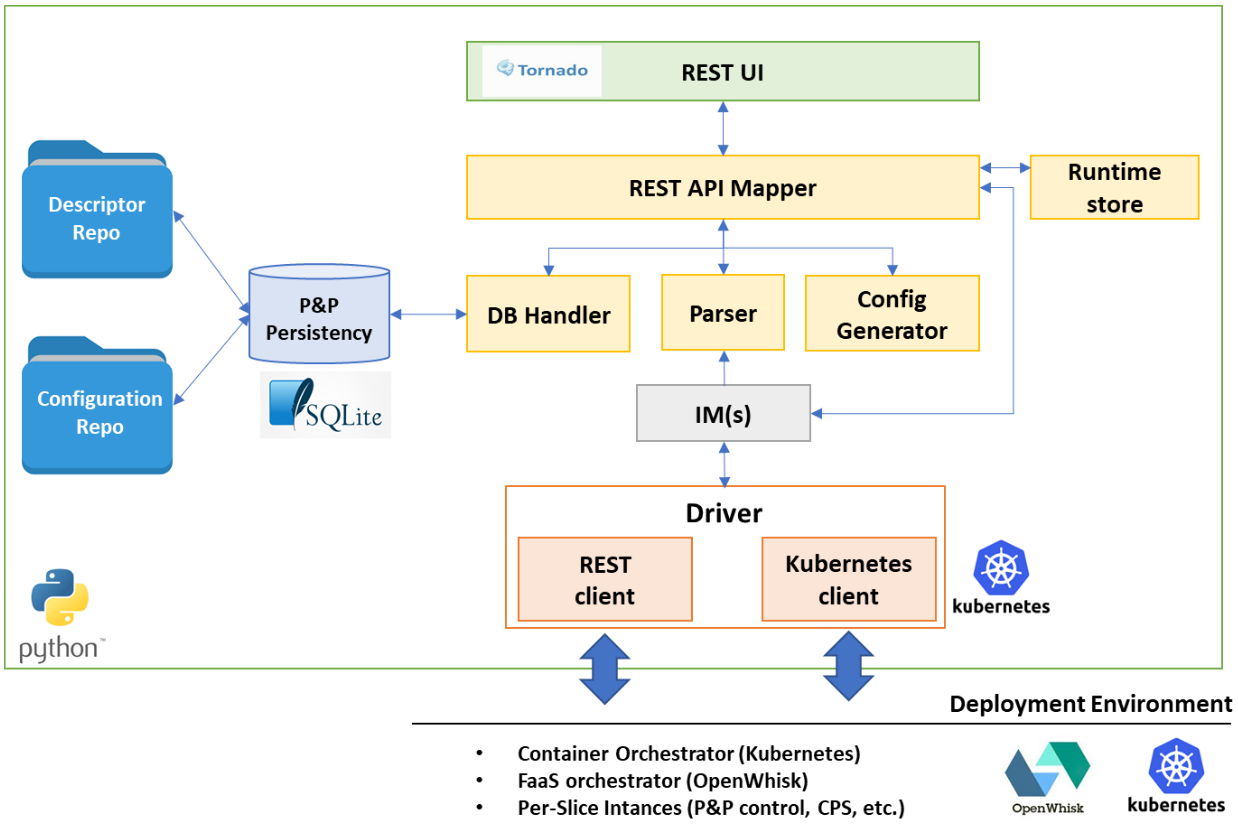


Figure 14 Software architecture of P&P Manager prototype

The architecture of the P&P Manager software prototype is depicted in Figure 14 and shows, exploiting the same combination of colours of the high-level architecture in Figure 9, in order to ease the mapping of the real software modules on the functional ones. This section presents the full P&P manager prototype architecture, detailing all of the modules in terms of functionalities, roles, implantation choices and of software frameworks.

All of the components are written in Python (v3.6) and the software is released as open source under the *Apache 2.0* license [31]. The P&P Manager source code is available on the SliceNet public gitlab project [32].

The ***REST UI*** is the component implementing the P&P Coordinator. In terms of functionalities, the module offers a set of REST APIs as defined in Section 4.3.1.1. From an implementation point of view, the main role of the REST UI is the one already described in Section 4.1 for the *P&P Coordinator*: offer an interface to access the P&P Manager functionalities. The base logic has been kept as simple as possible: the REST UI just handle the HTTP requests leaving any further elaboration to the modules implementing the P&P LCM.

The current implementation of the P&P LCM consists of several modules interacting together in order to realize the P&P Manager core features:

- Elaboration of the SS-O deployment requests;

- Access and manipulation of the P&P stored data (P&P Descriptor Catalogue and P&P Inventory);
- Requests/Responses parsing;
- Generation of the per-slice application instances configurations;
- Management of the P&P internal status;

The implementation of the REST UI is based on Tornado [33], a python Web Framework that offers a complete set of libraries for implement all of the features requested. In the current approach, the set of APIs implemented for the P&P REST UI are based on synchronous HTTP mechanism. In this way, the API caller, mainly the SS-O, should wait until the competition of the requested operation and then receive a reply from the P&P Manager. Future enhancements or migration towards an asynchronous set of REST APIs are under evaluation as part of the integration with the SS-O components. Indeed, Tornado provide a full support for asynchronous HTTP API can be exploited, as for P&P control described in D4.1, for non-blocking operations on the P&P manager interface. In this sense, the asynchronous interface should implement a two-phases mechanism for replying to the request of the SS-O. In the first phase, the P&P Manager should reply using simple HTTP Codes (e.g. 200 OK) in order to inform the SS-O that its own request has been (or a has not been) taken in charge. The second phase, doable only if the first one succeeded, should advertise the SS-O that the requested operation is now concluded (with success or not) and should provide any additional information needed by SS-O (e.g. how reach the interface of just created P&P control instance). In this regard, the SS-O should provide an endpoint, as part of its own interface, the P&P manager will use for pushing the information as depicted in Figure 15.

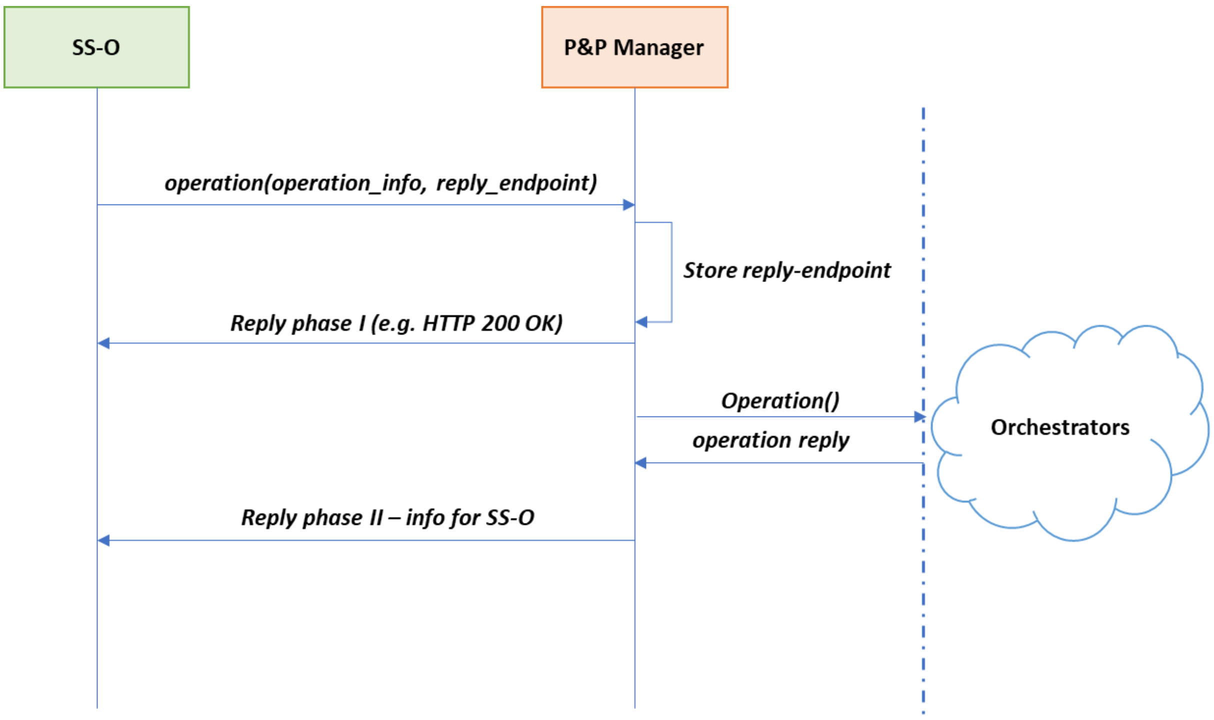

SS-O
P&P Manager
operation(operation_info, reply_endpoint)
Store reply-endpoint
Reply phase I (e.g. HTTP 200 OK)
Operation()
operation reply
Orchestrators
Reply phase II – info for SS-O

Figure 15 – Workflow of asynchronous interaction with SS-O

The component in charge of managing the request received from the REST UI is the ***REST API Mapper***. It implements a set of APIs that can be mapped one by one to those offered by the REST UI, along with a set of functions for internal use. As shown in Figure 14, the REST API mapper represents the central component of the P&P LCM implementing the actual logic needed for managing the SS-O requests.

Indeed, each other components belonging to P&P LCM is invoked only by the REST API Mapper that, combining the services offered by the single modules, is able to handle the complex requests received from the REST UI.

The information related to the actual status of the P&P Manager is managed by the ***Runtime store.*** It offers an interface for read/store/update/delete information about the modules created at runtime for handling a given set of per-slice application instances, the P&P IMs and, in addition, it stores all of those information that are not configuration or descriptor and are not stored in the Catalogue and in the Inventory residing in the persistent memory.

The ***Parser***, as the name suggests, is the module in charge of parsing all messages received by the P&P Manager. Such messages can be originated by the SS-O (e.g. the payload of a CREATE request) or can be sent by the orchestrators (*OpenWhisk* for FaaS, *Kubernetes* for containerized applications) as reply for the requests from P&P manager to their own interfaces. For that reason, such messages can have different natures and meanings and need of a proper module for parse, group and retrieve relevant information. The format used is JSON.

Once REST API Mapper retrieves all of the information needed for deploy a new set of per-slice application instances, it sends such information to the ***Config Generator***, by invoking proper APIs on its own interface, in order to:

- Produce request files needed by Kubernetes and/or OpenWhisk for creating new instance of the respective components
- Produce configuration files for the per-slice applications

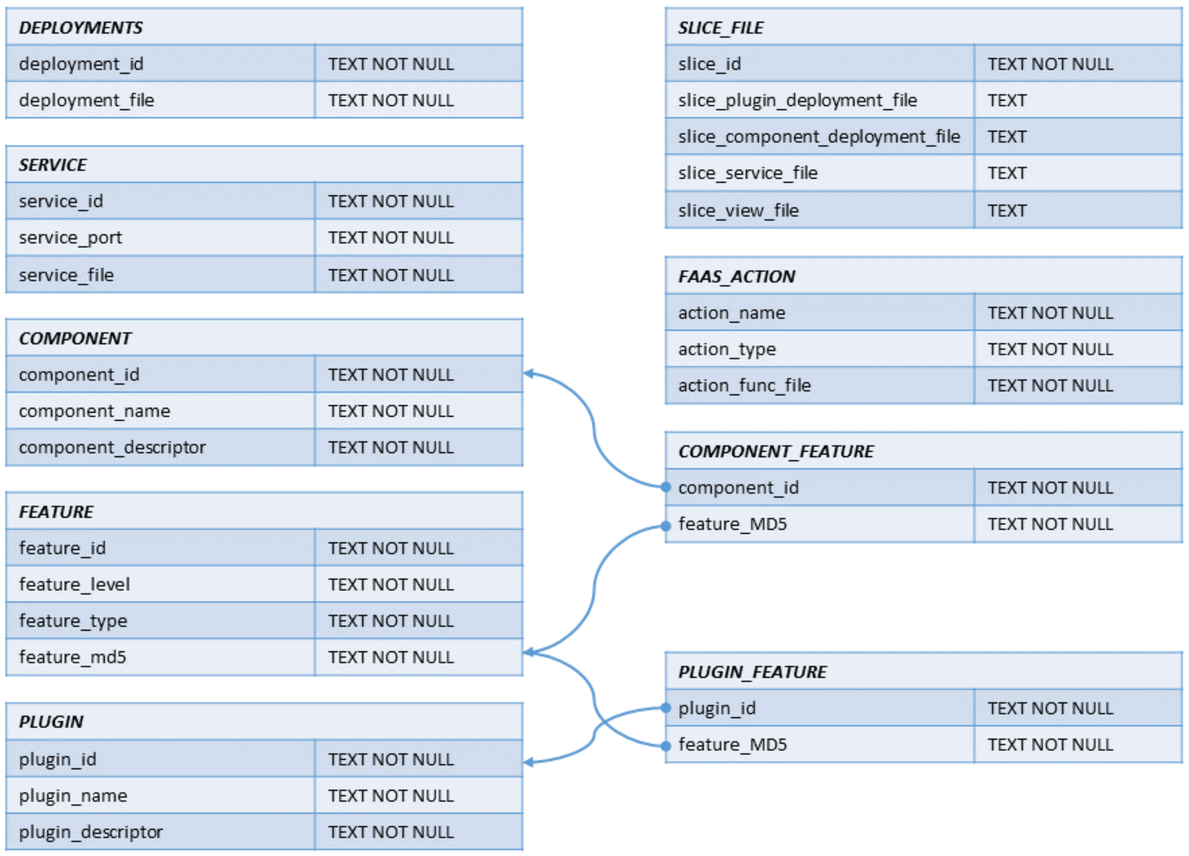
The produced files are stored in the Inventory and can be of two different formats, depending on the entity they will be processed from: YAML or JSON. The generation is based on proper templates stored in the Catalogue.

The P&P Inventory and the P&P Descriptor Catalogue are the two P&P Manager storage points that contains configuration and deployment files (typically JSON or YAML) and component Descriptors respectively. The implementation of such stores consists of two folders improved by an “augmented” indexing of the object files, based on an instance of SQLite [34] that contains useful information for quickly retrieve the Descriptor, Configuration or deployment file of interest.

The are several reasons why SQLite has been chosen for implementing the persistent storages of the P&P Manager. The first is given by the typology of data stored in both the P&P Inventory and in the P&P Configuration manager: they are files, JSON or YAML, arbitrarily long. It means that the content of such files may vary, even if the files belongs to the same category (e.g. Component Descriptors), making the storage of the data they contain complex in a common SQL-like database. An additional aspect to take into account is that only few information contained in the object files are suitable for the P&P Manager, the rest of the dada they stored is needed by other entities (e.g. Kubernetes). For that reason, the base idea is to store all of the information the P&P Manger uses for its own operation in a database, in order to quickly retrieve the files of interests by using the values contained in the database as search keys. In this sense, SQLlite fits perfectly.

SQLlite is a software library (it is written in C but wrappers in many programming languages, like Python, are available) that can be fully integrated in the software of the P&P Manager. It offers an SQL-like query interface that can be exploit for quickly match the desired features available in the P&P Manager stores with the ones requested by the SS-O. It is light: as software library, it is part of the P&P manager itself and works without create a separate process for managing the DB as other Database Management System do (e.g. MySQL), it is highly portable and released without any kind of copyright.

Figure 16 shows the set of tables composing the SQLite database.



DEPLOYMENTS	
deployment_id	TEXT NOT NULL
deployment_file	TEXT NOT NULL

SERVICE	
service_id	TEXT NOT NULL
service_port	TEXT NOT NULL
service_file	TEXT NOT NULL

COMPONENT	
component_id	TEXT NOT NULL
component_name	TEXT NOT NULL
component_descriptor	TEXT NOT NULL

FEATURE	
feature_id	TEXT NOT NULL
feature_level	TEXT NOT NULL
feature_type	TEXT NOT NULL
feature_md5	TEXT NOT NULL

PLUGIN	
plugin_id	TEXT NOT NULL
plugin_name	TEXT NOT NULL
plugin_descriptor	TEXT NOT NULL

SLICE_FILE	
slice_id	TEXT NOT NULL
slice_plugin_deployment_file	TEXT
slice_component_deployment_file	TEXT
slice_service_file	TEXT
slice_view_file	TEXT

FAAS_ACTION	
action_name	TEXT NOT NULL
action_type	TEXT NOT NULL
action_func_file	TEXT NOT NULL

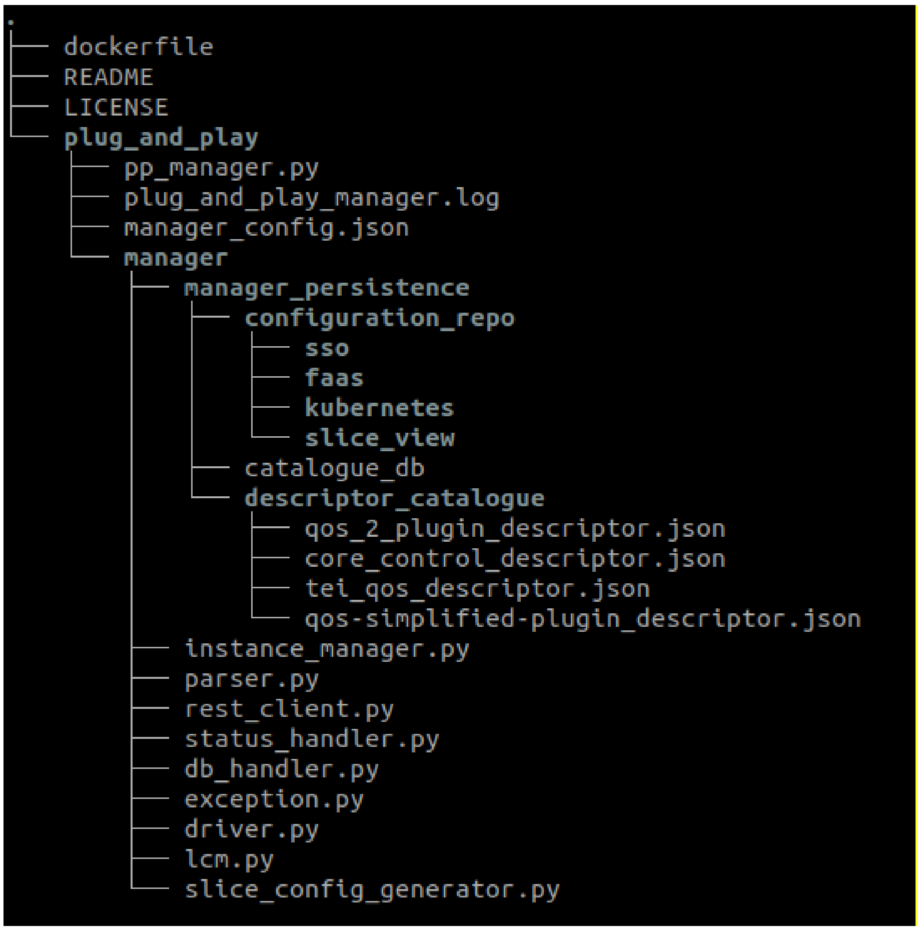
COMPONENT_FEATURE	
component_id	TEXT NOT NULL
feature_MD5	TEXT NOT NULL

PLUGIN_FEATURE	
plugin_id	TEXT NOT NULL
feature_MD5	TEXT NOT NULL

Figure 16 SQLite DB architecture

The P&P LCM accesses the persistent database through a proper module ***DB Handler*** that provides all of the API required for read, create, update and delete the indexed files. As described in Section 4.3.1.1, the P&P Manager implements an API for auto-(re)build the information contained in the database, basing on the files stored in each folder (Inventory and Catalogue).

The Inventory, in addition, contains sub-folders for grouping produced file for specific consumers: ***kubernetes***, for all of files needed for create instances using Kubernetes as orchestrator, ***faas***, which contains the source code files of the function to be uploaded on OpenWhisk and, finally, ***slice_config***, in which resides the per-slice applicaiton configurations. The original request, sent by the SS-O, is also stored under ***sso*** folder. The folder tree is shown in Figure 17:



```
.
├── dockerfile
├── README
├── LICENSE
└── plug_and_play
    ├── pp_manager.py
    ├── plug_and_play_manager.log
    ├── manager_config.json
    └── manager
        ├── manager_persistence
        │   ├── configuration_repo
        │   │   ├── sso
        │   │   ├── faas
        │   │   ├── kubernetes
        │   │   └── slice_view
        │   ├── catalogue_db
        │   └── descriptor_catalogue
        │       ├── qos_2_plugin_descriptor.json
        │       ├── core_control_descriptor.json
        │       ├── tei_qos_descriptor.json
        │       └── qos-simplified-plugin_descriptor.json
        ├── instance_manager.py
        ├── parser.py
        ├── rest_client.py
        ├── status_handler.py
        ├── db_handler.py
        ├── exception.py
        ├── driver.py
        ├── lcm.py
        └── slice_config_generator.py
```

Figure 17 Tree representation of the P&P module files.

After the request from the SS-O has been elaborated and all of the deployment file is ready to be applied, the P&P LCM, through the P&P REST API mapper, as first action, creates an instance of the ***P&P Instance Manager*** (P&P IM), for i) enable communication with the Container Orchestrators and ii) after slice deployment, manage the instances through a proper API interface. Indeed, the P&P IM(s) interacts directly with the P&P Driver. Furthermore, each P&P IM keeps the status information of each set of components it has instantiated like deployment IDs, service IDs, etc. After they have been created, P&P IMs are stored in a map residing on the Runtime Store and indexed per slice_id, in order to make the P&P LCM able to recall the proper IM every time SS-O invoke a per-slice operation involving the instances or the orchestrators, like *UPDATE* or *TERMINATE* or *CONFIGURE*.

The ***P&P Driver*** is realized as a couple of clients being able to access both the Kubernetes and OpenWhisk as well as the P&P Control instances for configuration. In particular, Kubernetes provides a Python package [35] implementing a set of APIs for dealing with the ones exposed by the *kube-apiserver*. The *kube-apiserver* is a process that resides in the Kubernetes Master Node (main node of a Kubernetes cluster) and exposes the set of REST APIs for a complete control of a Kubernetes environment. The second module client is a REST client based on Python Requests library [36]. Such client is used both for configure P&P Control instances and for interacting with OpenWhisk.

Tool Name and Version	Description
Python 3.6	All P&P Manager core modules developed, including the ones relying on particular packages, are compatible with Python version 3.6
Tornado 5.1	It is the Python web application framework used to implement the P&P Manager REST UI (representing the implementation of P&P Coordinator)
Requests 2.19.1	It is the Python HTTP library used to implement the REST Client at P&P Manager south-bound interface
Kubernetes-client 9.0.0	It is a Python module implementing a client for accessing the Kubernetes API Server

Table 19 P&P Manager software dependencies

5.1 Preliminary integration and validation

The P&P Manager has been preliminary tested and validated in a controlled testbed built in the Nextworks lab.

The scenario

As for the P&P Control, the validation context is the Smart Grid UC. The validation scenario consists of two IEDs (Intelligent Electronic Device) connected via 5G/4G-LTE to a SCADA (Supervisory Control And Data Acquisition) application. Such elements are not actually deployed and are simulated in the testbed environment. Indeed, for the purpose of the demonstration, we do not need to have the real smart grid devices in place: they are simulated for giving a context to the demo and generating a slice view for the One-Stop-API GUI. The demo workflow consists of 3 main steps: i) create per-slice applications request, ii) deployment of the per-slice applications and, finally, iii) interaction with the deployed applications through the One-stop-API GUI.

The requested slice will consist of one instance of the P&P control (as per D4.1) and one of QoS control plane service (as per D4.3). Such instance of P&P Control will deploy, along with the main abstraction P&P service, two different P&P functions. One of them, the QoS monitoring plugin, is a stub that generates random throughput value (as a way to emulate IEDs and SCADA that are not actually deployed) while the other one, the QoS control plugin, is in charge of interacting with QoS control plane service. Such interaction is kept simple for testing purposes: the QoS control plugin performs a GET on the QoS control plane service REST interface in order to retrieve its base configuration information.

It is worth noting that in this scenario, the P&P Manager is not distinguishing between DSP and NSP: the P&P Control instance, that build the slice view for the vertical, and the QOS-CP, that is a CPSs and resides at NSPs only, are deployed in the same environment. Indeed, this deployment has to be intended as a pure integration scenario aiming at validating the capacity of the P&P Manager of processing a *create per-slice applications request* from the SS-O, selecting the right applications, deploying and configuring them.

Figure 18 shows a representation of the slice as it would be represented by OSA GUI. The two IEDs are connected to a SCADA node (*CTRL-001*) via 5G and belonging to a hypothetical *edge_aveiro.* The isolated node on the left represents the slice itself, with its name *smart-grid-001* and *idbf5cbb04-345c-479f-8511-f6f01b2b822d*. As discussed in D4.1, such node was used for associating slice overall features. In this case, as described below, we assume that the P&P QoS Monitoring plugin offers a general information about the throughput of the slice so, its functionalities will be associated to the slice node while, the QoS control plugin, in charge of interacting with QoS control plane service will be associated to the SCADA node in the topology view.

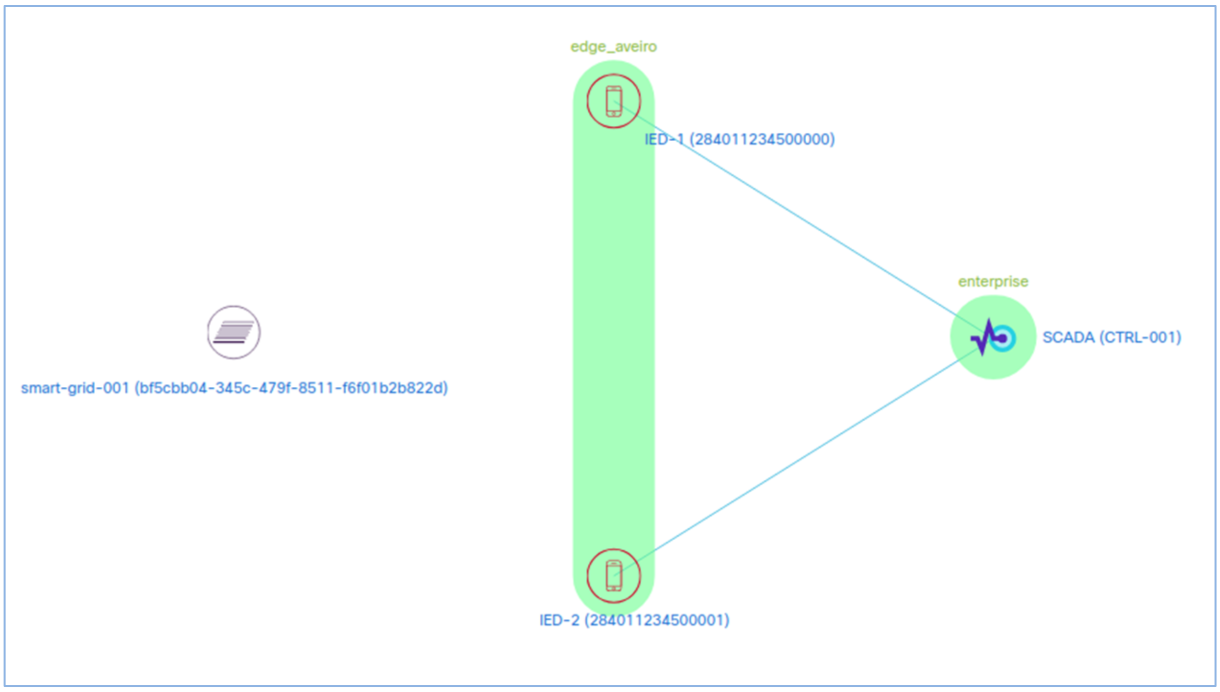


Figure 18 – View of the Demo scenario from the OSA GUI

Testbed environment

The testbed consists of two nodes as depicted in Figure 19. The first nodes is a physical machine (a laptop), in which are deployed an instance of P&P Manager along with an instance of Postman [37], the tool for testing REST interfaces. In this case, Postman emulates the requests that should issues by the SS-O. The second node is a Virtual Machine running in an OpenStack [38], environment and hosting an instance of the Control Plane Service Registry (CPSR [25]). The VM is also configured as single-node cluster for Kubernetes and is used for deploying the per-slice application instances.

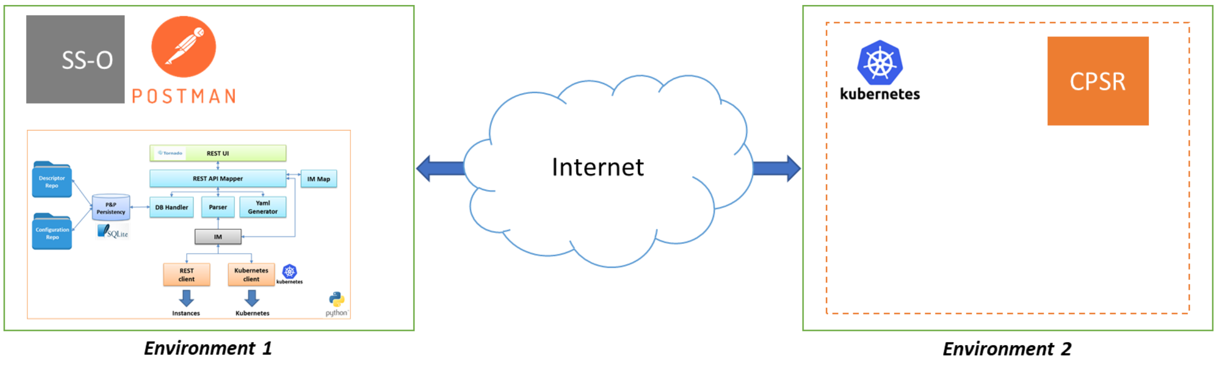


Figure 19 Demo testbed at Nextworks Lab

Step 1 (Figure 20) *– Create a new set of per-slice application instances.*

1. The SS-O (Postman) sends a slice *CREATE* request against the P&P Manager interface along with the proper request payload (see Annex A).

2. P&P Manager processes the request and creates proper configuration files for Kubernetes. Then it sends them to the Kubernetes Cluster through the Internet.
3. As result of the P&P Manager request, Kubernetes creates 2 Pods: 1 for the P&P control instance and one for the QoS control plane service. The P&P control instance consist of 3 containers: the core abstraction (blue), the QoS Monitoring plugin (grey) and the QoS Control plugin (green)
4. After creation, the QoS control plane service registers itself to the CPSR, while the P&P Control is exposed as Kubernetes service (indicated as a grey circle in Figure 20)

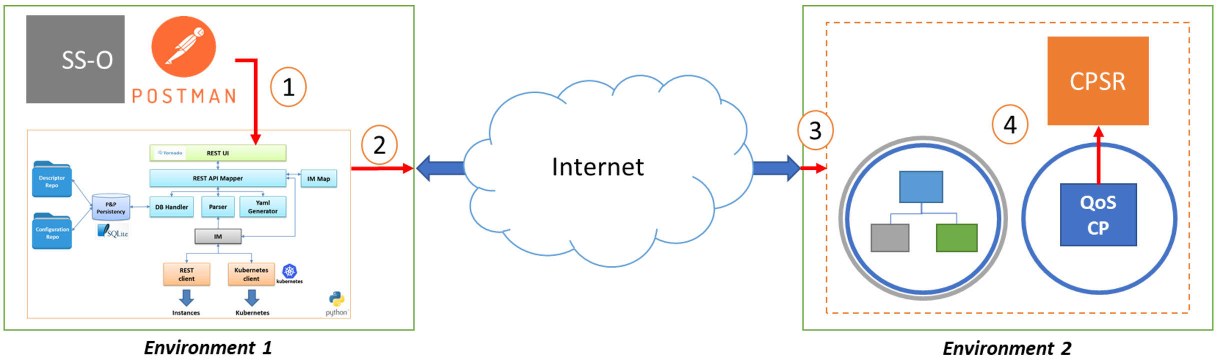


Figure 20 Demo Step 1: deployment of the requested instances, P&C control exposition as Kubernetes service and QoS-CP registration to CPSR

Figure 21 shows the results of two Kubernetes CLI commands. The first, *get pods* lists the Kubernetes pods created at slice creation, named basing on the following naming structure:

<slice-id>-<type>-<kuberntes-pod-id>

In this case, there are two pods: one for the QoS control plane service (type: component) and one for the P&P control instance (type: plugin). The numbers in the "READY" column represent the number of containers in the pod that are running: 3 (of 3) for the P&P Control, the core and the two plugins and 1 (of 1) for the QoS-CP.

The second Kubernetes CLI commands show the list of the running Kubernetes service. The P&P Control is exposed as Kubernetes service and indicated with the slice id: *idbf5cbb04-345c-479f-8511-f6f01b2b822d.*

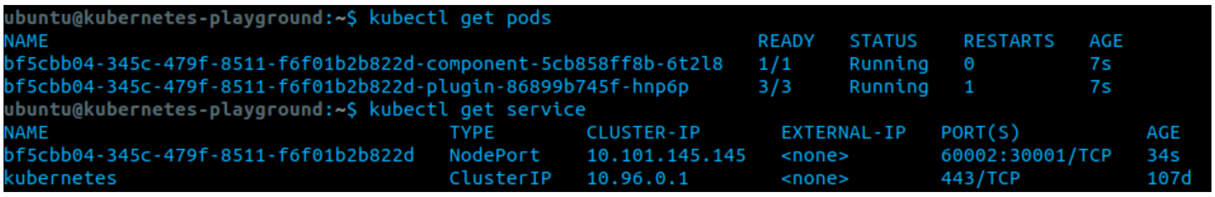
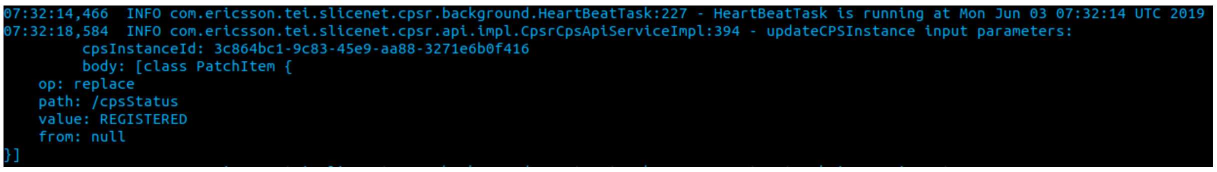

```
ubuntu@kubernetes-playground:~$ kubectl get pods
NAME                                                    READY   STATUS    RESTARTS   AGE
bf5cbb04-345c-479f-8511-f6f01b2b822d-component-5cb858ff8b-6t2l8   1/1     Running   0          7s
bf5cbb04-345c-479f-8511-f6f01b2b822d-plugin-86899b745f-hnp6p      3/3     Running   1          7s
ubuntu@kubernetes-playground:~$ kubectl get service
NAME                                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)           AGE
bf5cbb04-345c-479f-8511-f6f01b2b822d    NodePort    10.101.145.145   <none>        60002:30001/TCP   34s
kubernetes                              ClusterIP   10.96.0.1        <none>        443/TCP           107d
```

Figure 21 Kubernetes CLI commands confirming the container deployment and service creation

Figure 22 reports a piece of the log of the CPSR showing that the QoS contorl plane service has been succeefully registered with id: *3c864bc1-9c83-45e9-aa88-3271e6b0f416*. The same id will be reported on the One-Stop-API GUI once the QoS control plane service will be queried.



```
07:32:14,466  INFO com.ericsson.tei.slicenet.cpsr.background.HeartBeatTask:227 - HeartBeatTask is running at Mon Jun 03 07:32:14 UTC 2019
07:32:18,584  INFO com.ericsson.tei.slicenet.cpsr.api.impl.CpsrCpsApiServiceImpl:394 - updateCPSInstance input parameters:
        cpsInstanceId: 3c864bc1-9c83-45e9-aa88-3271e6b0f416
        body: [class PatchItem {
    op: replace
    path: /cpsStatus
    value: REGISTERED
    from: null
}]
```

Figure 22 Screenshot of CPSR logs evidencing the QoS CP registration

Step 2 (Figure 23) – Configuration of the P&P Control instance: The vertical-tailored Slice View

1. SS-O (Postman) sends a configure slice request on P&P manager
2. P&P Manager processes the request and creates proper slice topology view for the P&P Control instance. Then it sends it as configuration to the P&P control instance running in Environment 2
3. P&P Control Core receives the configuration

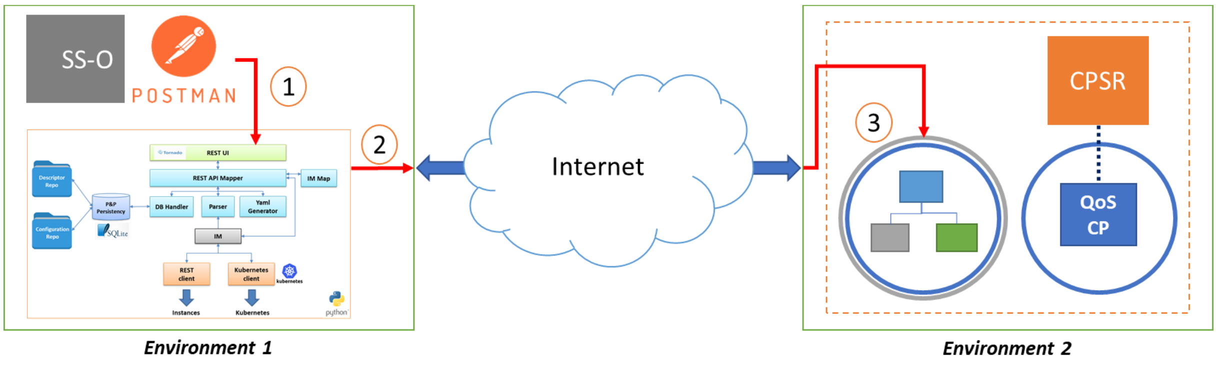


Figure 23 Demo Step 2: The P&P control instance is configured with the slice view

Step 3 (Figure 24) - Interaction with Slice Instance through the One-Stop-API GUI – Example of a request towards QoS control plane service

1. One-Stop-API GUI uses the APIs generated by P&P Control Instance interface for requesting QoS control plane instance info
2. P&P Control Core receives the request and forward it towards the proper plugin (QoS Control)
3. The QoS control plugin requests the QoS control plane service address to CPSR
4. Finally, the QoS control plugin requests the information to QoS control plane service

As a result visible on the One-Stop-API GUI, the QoS control plane service information is forwarded towards back to the GUI.

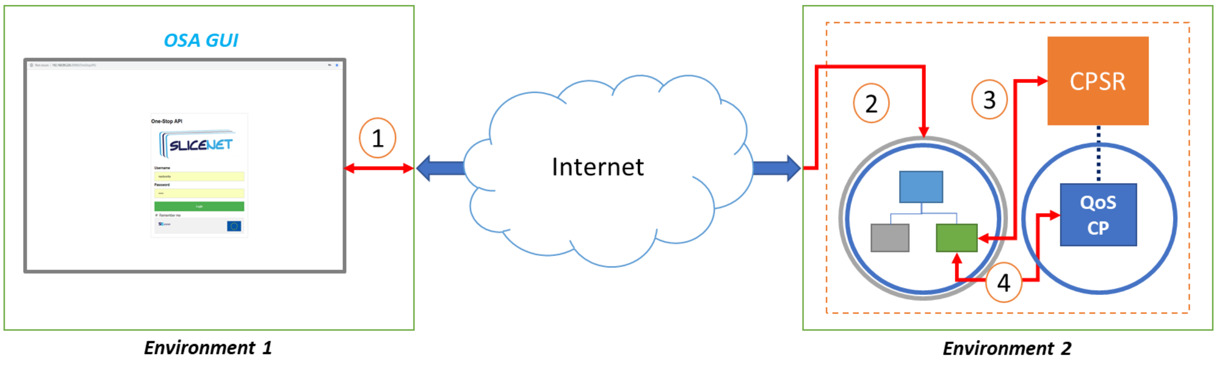
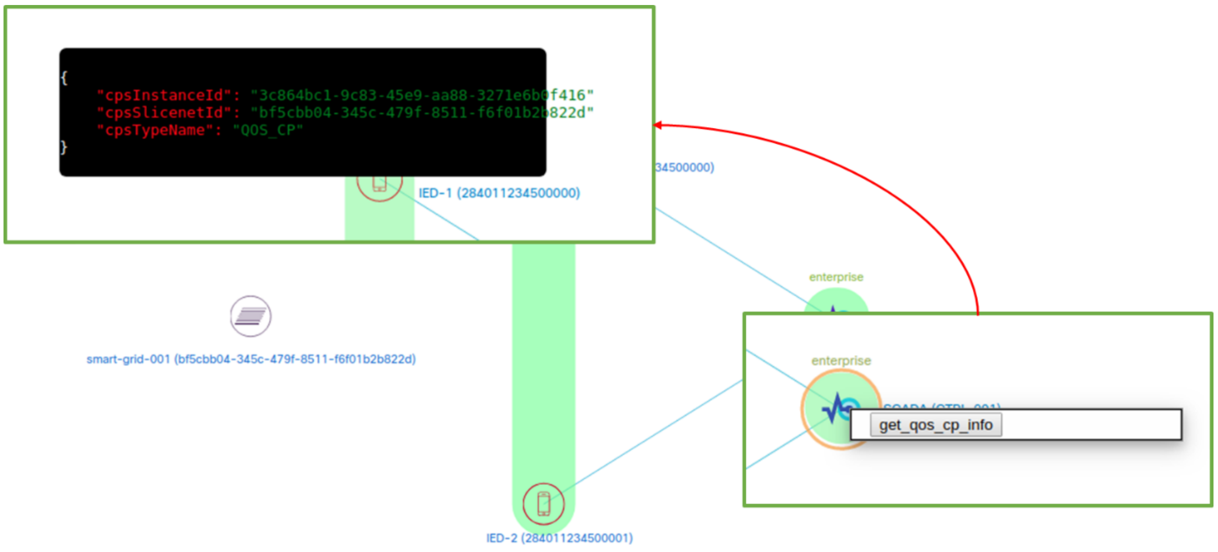


Figure 24 Demo Step 3: Sequence of actions for retrieving information from QoS-CP

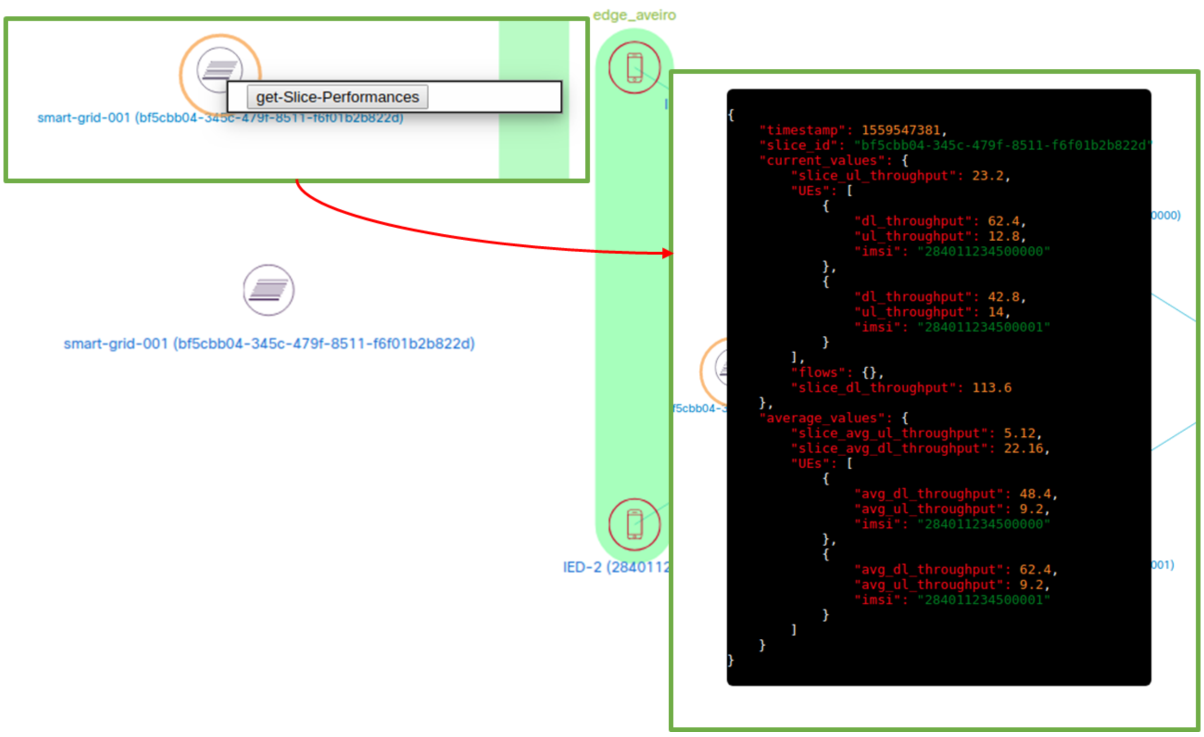
The results of the step 3 are shown in Figure 25. The feature “get_qos_cp_info” is exposed by the SCADA node and, by clicking the button, the requested QoS-CP information retrieved appear on the screen: the *slice id*, the *cps instance id* (id of the QoS-CP instance), and the *type* of the CPS.



```
{
    "cpsInstanceId": "3c864bc1-9c83-45e9-aa88-3271e6b0f416"
    "cpsSlicenetId": "bf5cbb04-345c-479f-8511-f6f01b2b822d"
    "cpsTypeName": "QOS_CP"
}
```

Figure 25 QoS-CP information retrieving from One-Stop-API GUI

For sake of completeness, Figure 26 shows the throughput information retrieved by exploiting the *get-Slice-Performance* feature exposed by the slice node. As discussed above, the data are not real and have been generated by the QoS Monitoring plugin.


get-Slice-Performances
smart-grid-001 (bf5cbb04-345c-479f-8511-f6f01b2b822d)
edge_aveiro
smart-grid-001 (bf5cbb04-345c-479f-8511-f6f01b2b822d)
IED-2 (2840112
0000)
f5cbb04-3
001)

```
{
    "timestamp": 1559547381,
    "slice_id": "bf5cbb04-345c-479f-8511-f6f01b2b822d"
    "current_values": {
        "slice_ul_throughput": 23.2,
        "UEs": [
            {
                "dl_throughput": 62.4,
                "ul_throughput": 12.8,
                "imsi": "284011234500000"
            },
            {
                "dl_throughput": 42.8,
                "ul_throughput": 14,
                "imsi": "284011234500001"
            }
        ],
        "flows": {},
        "slice_dl_throughput": 113.6
    },
    "average_values": {
        "slice_avg_ul_throughput": 5.12,
        "slice_avg_dl_throughput": 22.16,
        "UEs": [
            {
                "avg_dl_throughput": 48.4,
                "avg_ul_throughput": 9.2,
                "imsi": "284011234500000"
            },
            {
                "avg_dl_throughput": 62.4,
                "avg_ul_throughput": 9.2,
                "imsi": "284011234500001"
            }
        ]
    }
}
```

Figure 26 Overall slice throughput retrieving from OSA GUI

6 Conclusions

This deliverable has presented the design and implementation of the P&P Manager component within the SliceNet system architecture. The P&P Manager is the lifecycle, and resides within the SliceNet orchestration sub-plane, at both DSP and NSP levels, even if with different scopes and coordination logics. The original scope of the SliceNet P&P approach has been enhanced towards a more comprehensive paradigm for enabling the deployment of per-slice control and management applications within the SliceNet cognitive management platform, to be integrated with classical monolithic cross-slice orchestration and management system functions. This approach follows recent trends in the telco industry for having truly programmable and flexible network and service management architectures, able to cope with the high dynamicity and heterogeneous requirements imposed by vertical industry players as well by 5G networks and services.

The full P&P Manager architecture design is presented in this deliverable, along with the definition of relevant information models and external interfaces APIs that drive its interaction with other components of the SliceNet system architecture. The P&P Manager has been completely designed and implemented from scratch, following an incremental approach to cover the initial P&P scope defined in D2.3 and D4.1 first, and the new features for programmable and per-slice control and management customized features introduced with this document.

A software prototype for the SliceNet P&P Manager has been also fully developed and presented in the document, and it has been produced relying on open source Python frameworks as Tornado and Requests as well as on software packages properly developed for interacting with Kubernetes and OpenWhisk as container orchestrators for running the per-slice control and management applications. A preliminary consistent integration effort has been also put and reported in the document, trying to have an initial assessment of the evolved P&P approach, comprising lifecycle management and dynamic and on-demand control features. In this scope, the integration with OpenWhisk and Kubernetes has been already tested (even if in a controlled environment) and can be considered in a mature phase.

The testing and validation efforts will continue by integrating the P&P Manager with other components in the SliceNet system architecture, like the SS-O and the One-Stop-API service access layer. The integration with these two components is crucial at both DSP and NSP levels for addressing the challenge of deploying and exposing a set of per-slice control and management applications in a scalable, transparent and programmable way.

Annex A

Example of deployment info model for the demo discussed in Section 4.3.

```
{
 "slice_id": "bf5cbb04-345c-479f-8511-f6f01b2b822d",
 "slice_name": "smart-grid-001",
 "slice_owner": "efacec",
 "slice_domain_type": "single",
 "required_feature":[
   {
     "seq_id": "0",
     "feature_id":  "qos_ctrl_kpi",
       "feature_type": "management",
       "feature_level": "element"
     },
     {
       "seq_id": "1",
       "feature_id":  "qos_mon_kpi",
       "feature_type": "monitoring",
       "feature_level": "slice"
     }
   ],
   "required_faas":[
     "make_greetings"
   ],
   "domains" : [
     {
       "domain_id": "domain-001",
       "elements": [
         {
           "id": "bf5cbb04-345c-479f-8511-f6f01b2b822d",
           "name": "smart-grid-001",
           "type": "node",
           "features": [
             "1"
           ]
         },
         {
           "id": "CTRL-001",
           "name": "SCADA",
           "type": "node",
           "features": [
             "0"
           ]
         },
         {
           "id": "284011234500000",
           "name": "IED-1",
           "type": "node",
           "features": [

           ]
         },
         {
           "id": "284011234500002",
           "name": "IED-2",
           "type": "node",
           "features": [

           ]
         }
       ],
       "connections": [
         {
           "id": "CTRL-001",
           "adjacent_elements": [
             "284011234500000",
             "284011234500001"
           ]
         }
       ]
     }
   ]
 }
```

References

[1] SliceNet D2.3, "Control Plane System Definition, APIs and Interfaces", SliceNet Consortium, April 2018

[2] SliceNet D2.4, "Management Plane System Definition, APIs and Interfaces", May 2018

[3] SliceNet D4.1, "Plug & Play Control Plane for Sliced Networks", October 2018

[4] 5GPPP Phase 2 – 5G Transformer, http://5g-transformer.eu/

[5] 5GPPP Phase 2 – MATILDA, http://www.matilda-5g.eu/

[6] 5GPPP Phase 2 – 5GCity, https://www.5gcity.eu

[7] 5GPPP Phase 2 – 5G ESSENCE, www.5g-essence-h2020.eu

[8] 5GPPP Phase 2 – 5G MEDIA, www.5gmedia.eu

[9] 5GPP Phase 3 – 5G EVE, https://www.5g-eve.eu/

[10] 5G EVE D3.1 "Interworking capability definition and gap analysis document", December 2018

[11] 5GPP Phase 3 – 5G VINNI, https://www.5g-vinni.eu/

[12] 5G VINNI D1.1, "Design of infrastructure architecture and subsystems v1", December 2018

[13]5GPP Phase 3 – 5Genesis, https://5genesis.eu/

[14] 5Genesis D2.2, "Initial overall facility design and specifications", November 2018

[15] OSM, https://osm.etsi.org/

[16] OSM Release 5, https://osm.etsi.org/wikipub/index.php/OSM_Release_FIVE_Documentation

[17] ONAP, https://www.onap.org/

[18] ONAP Release 4, https://wiki.onap.org/display/DW/Dublin+Architecture

[19] Simon Dredge, "5G Future Begins with Cloud Native NFV Today", Aug 2, 2018, https://www.metaswitch.com/blog/5g-future-begins-with-cloud-native-nfv-today

[20] 5G-PPP Software Network Working Group, "From Webscale to Telco, the Cloud Native Journey", July 2018.

[21] Ericsson 5G Core solution, https://www.ericsson.com/en/digital-services/offerings/core-network/5g-core

[22] ETSI ZSM ISG, https://www.etsi.org/technologies/zero-touch-network-service-management

[23] ETSI ZSM-002 " Zero-touch network and Service Management (ZSM); Reference Architecture", https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=54295

[24] SliceNet D2.2, "Overall Architecture and Interfaces Definition", SliceNet Consortium, January 2018

[25] SliceNet D4.3 "Single-Domain, Multi-Tenant Network Slicing Control", December 2018

[26] SliceNet D2.4, "Management Plane System Definition, APIs and Interfaces", May 2018

[27] Kubernetes, https://kubernetes.io/

[28] Amazon Lambda, https://aws.amazon.com/lambda/?nc1=f_ls

[29] OpenWhisk, https://openwhisk.apache.org/

[30] SliceNet D6.6 "Single-Domain Slice FCAPS management", May 2019

[31] Apache 2.0 License, https://www.apache.org/licenses/LICENSE-2.0.html

[32] Plug & Play Manager software repository, https://gitlab.com/slicenet/plug-and-play-manager

[33] Tornado Web Server, https://www.tornadoweb.org/en/stable/

[34] SQLite, https://www.sqlite.org/index.html

[35] Kubernetes Python client, https://github.com/kubernetes-client/python

[36] Python Requests, https://2.python-requests.org/en/master/

[37] Postman, https://www.getpostman.com/

[38] OpenStack, http://www.openstack.org